

MAGAZINE

BSD

FOR NOVICE AND ADVANCED USERS

FREEBSD PROGRAMMING PRIMER

NETBEANS AND XDEBUG

CASCADING STYLE SHEETS

JAVASCRIPT

JQUERY

MYSQL

12 tutorials
How to configure
a development
environment and
write HTML, CSS,
PHP and SQL code

VOL.3 NO.1
ISSUE 01/2014(4)
1898-9144



855-GREP-4-IX
www.iXsystems.com
Enterprise Servers and Storage
for Open Source



- ✓ Rock-Solid Performance
- ✓ Professional In-House Support

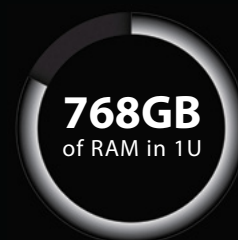
E5-2600

High Performance, High Density Servers for Data Center, Virtualization, & HPC



MODEL: iXR-22X4IB

<http://www.iXsystems.com/e5>



KEY FEATURES

iXR-22X4IB

- Dual Intel® Xeon® Processors E5-2600 Family per node
- Intel® C600 series chipset
- Four server nodes in 2U of rack space
- Up to 256GB main memory per server node
- One Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector per node
- 12 SAS/SATA drive bays, 3 per node
- Hardware RAID via LSI2108 controller
- Shared 1620W redundant high-efficiency Platinum level (91%+) power supplies

iXR-1204+10G

- Dual Intel® Xeon® Processors E5-2600 Family
- Intel® C600 series chipset
- Intel® X540 Dual-Port 10 Gigabit Ethernet Controllers
- Up to 16 Cores and 32 process threads
- Up to 768GB main memory
- Four SAS/SATA drive bays
- Onboard SATA RAID 0, 1, 5, and 10
- 700W high-efficiency redundant power supply with FC and PMBus (80%+ Gold Certified)

Call iXsystems toll free or visit our website today! **1-855-GREP-4-IX** | www.iXsystems.com

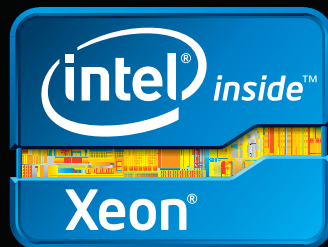
High-Density iXsystems Servers powered by the Intel® Xeon® Processor E5-2600 Family and Intel® C600 series chipset can pack up to 768GB of RAM into 1U of rack space or up to 8 processors - with up to 128 threads - in 2U.

On-board 10 Gigabit Ethernet and Infiniband for Greater Throughput in less Rack Space.

Servers from iXsystems based on the Intel® Xeon® Processor E5-2600 Family feature high-throughput connections on the motherboard, saving critical expansion space. The Intel® C600 Series chipset supports up to 384GB of RAM per processor, allowing performance in a single server to reach new heights. This ensures that you're not paying for more than you need to achieve the performance you want.

The iXR-1204 +10G features dual onboard 10GigE + dual onboard 1GigE network controllers, up to 768GB of RAM and dual Intel® Xeon® Processors E5-2600 Family, freeing up critical expansion card space for application-specific hardware. The uncompromised performance and flexibility of the iXR-1204 +10G makes it suitable for clustering, high-traffic web servers, virtualization, and cloud computing applications - anywhere you need the most resources available.

For even greater performance density, the iXR-22X4IB squeezes four server nodes into two units of rack space, each with dual Intel® Xeon® Processors E5-2600 Family, up to 256GB of RAM, and an on-board Mellanox® ConnectX QDR 40Gbp/s Infiniband w/QSFP Connector. The iXR-22X4IB is perfect for high-powered computing, virtualization, or business intelligence applications that require the computing power of the Intel® Xeon® Processor E5-2600 Family and the high throughput of Infiniband.



iXR-1204+10G: 10GbE On-Board



iXR-22X4IB

Intel, the Intel logo, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the U.S. and other countries.

Call iXsystems toll free or visit our website today! 1-855-GREP-4-IX | www.iXsystems.com

FreeBSD Programming Primer

12 tutorials

by Rob Somerville

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.



freeBSD

Editor in Chief:

Ewa Dudzic
ewa.dudzic@software.com.pl

Contributing:

Michael Shirk, Andrey Vedikhin, Petr Topiarz,
Charles Rapenne, Anton Borisev

Top Betatesters & Proofreaders:

Annie Zhang, Denise Ebery, Eric Geissinger, Luca Ferrari,
Imad Soltani, Olaoluwa Omokanwaye, Radjis Mahangoe,
Mani Kanth, Ben Milman

Special Thanks:

Annie Zhang
Denise Ebery

Art Director:

Ireneusz Pogroszewski

DTP:

Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Senior Consultant/Publisher:

Pawel Marciniak
pawel@software.com.pl

CEO:

Ewa Dudzic
ewa.dudzic@software.com.pl

Production Director:

Andrzej Kuca
andrzej.kuca@software.com.pl

Publisher:

Hakin9 Media SK
02-676 Warsaw, Poland
Postepu 17D
Poland
worldwide publishing
editors@bsdmag.org
www.bsdmag.org

Hakin9 Media SK is looking for partners from all over the world. If you are interested in cooperation with us, please contact us via e-mail: editors@bsdmag.org.

All trade marks presented in the magazine were used only for informative purposes. All rights to trade marks presented in the magazine are reserved by the companies which own them.

06 **FreeBSD Programming Primer – Part 1**
Rob Somerville

48 **FreeBSD Programming Primer – Part 7**
Rob Somerville

10 **FreeBSD Programming Primer – Part 2**
Rob Somerville

56 **FreeBSD Programming Primer – Part 8**
Rob Somerville

14 **FreeBSD Programming Primer – Part 3**
Rob Somerville

62 **FreeBSD Programming Primer – Part 9**
Rob Somerville

35 **FreeBSD Programming Primer – Part 4**
Rob Somerville

76 **FreeBSD Programming Primer – Part 10**
Rob Somerville

36 **FreeBSD Programming Primer – Part 5**
Rob Somerville

84 **FreeBSD Programming Primer – Part 11**
Rob Somerville

42 **FreeBSD Programming Primer – Part 6**
Rob Somerville

88 **FreeBSD Programming Primer – Part 12**
Rob Somerville

a d v e r t i s e m e n t



Web Based CRM & Business Applications for small and medium sized businesses

Find out how Workbooks CRM can help you

- Increase Sales
- Generate more Leads
- Increase Conversion Rates
- Maximise your Marketing ROI
- Improve Customer Retention

Contact Us to Find Out More

+44(0) 118 3030 100

info@workbooks.com



FreeBSD Programming Primer – Part 1

In this new series we will look at the tools, processes and methods involved in writing software, including developing a Content Management System (CMS) which will run under an AMP stack on FreeBSD, OpenBSD, Linux, etc.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills
-

Within the I.T. environment there are many disciplines, and often these skill sets work in isolation. The sys-admin doesn't always understand the challenges faced by the programmer or developer, the support engineer doesn't understand the problems of the developer, and the project manager doesn't understand the problems of the technical staff. In this new series, we will examine from first principles how to develop a CMS that will run on any Apache / MySQL / PHP stack. This will involve writing HTML, CSS, PHP and SQL code.

Code is Everywhere

To the uninitiated, writing computer code from scratch may seem a challenge. Certainly, some programming languages are more complex than others, but the fact remains you have already programmed some device at some stage without realizing it even if you have not been near the command line (for example a VHS recorder, central heating timer etc.). As a result you have instructed the device to do something (Record the Simpsons at 10:00PM on Friday evenings). Software is effectively just a collection of instructions, logic and actions like this that allow the computer to interact with another computer, an end user or just itself. The skill is in writing good code that meets the following guiding principles:

- Does “what it says on the tin”
- Is user friendly
- Is secure and reliable under stress
- Is fast and efficient (Don't Repeat Yourself)
- Is easily modified and extended
- Can be easily understood
- Has documentation

While some of these points are essential to any piece of software, some may be more important than others depending on the operating environment and specification. For instance, a piece of code that pulls pages from a website on a daily basis into a new directory in the format day_month_year (like 01_01_2013, 02_01_2013 etc.) for later reading by a technician would not necessarily require anything other than a log file entry saying “404 Not Found” if no content was available. However, if this was a critical program designed for an end user, it would be better practice to raise a friendly error message e.g. “The page you requested was not found. Please try again later or contact the helpdesk on 123 456789”.

Software writing should be creative and enjoyable, and part of the challenge is to have a reasonable idea of what you want to achieve beforehand, who your audience is, what limitations you must consider, and the environment the software will run under. A good functional

specification should cover these details, but it is important to realize that software is never really finished. More functionality may be required, the environment may change, or bugs and faults need to be rectified in the program. That is why code should be easily modified and understood as it is the programmers worst nightmare having to maintain a badly written, undocumented, broken program. Trying to get inside someone else's logic especially when under pressure to meet deadlines can be very stressful!

Computers Are Not Very Clever

The old adage “Garbage In = Garbage Out” is most applicable in the area of programming. As CANVC, they can only literally interpret any instructions that they receive. For instance, you might think you have asked the program to print the date, but due to an error in your logic, it might return 01-01-1970, NULL, or UNDEF. It might not even return anything at all. Sometimes when writing code you will be convinced the computer is your enemy. This is where defensive programming and debugging come to the fore, by re-thinking the obvious (and not so obvious) assumptions such as “All input data is valid”. The defensive programmer would respond by saying “All data is important and tainted unless proved otherwise”. Expect the unexpected. Sometimes it is best to walk away, take a break and return to the problem later. Late night coding sessions can be frustrating, especially if the result is not what is expected. Trying to debug an issue without a decent IDE (Integrated Development Environment) is possible, but time consuming.

Choosing the Language

Not all programming languages are equal, and some are less equal than others. Different languages are geared towards different tasks.

Shell programming languages (for example Bash, Sh etc.) are great for system administration tasks e.g. clearing out and archiving directories, running commands depending on the user response etc. However they are not fully fledged programming languages as such.

BASIC and Pascal are great for learning how to code, but they have some limitations. While it would be possible to write a CMS in either of them, as they are not primarily geared towards the web the program would be complex and convoluted.

The same argument applies to C. C is extremely powerful and flexible and PHP, Apache and MySQL are written using it. It would be complete overkill to write the CMS in scratch from C as we would effectively have to re-invent the wheel.

Java would make a great platform for a CMS due in part to its extensive library support and security, but as it is object orientated rather than procedural, the code and underlying principles would be more complex.

Script based languages (for example Ruby, Perl, Python, PHP) are geared towards the Internet, and most ISP's will support them. As PHP has good support, is very portable, the documentation is excellent, and integrates well with both Apache (Our web server software) and MySQL (our database) it is a strong choice. While the other script languages are just as suitable for our CMS, the author has more experience with PHP so that is the reason for the choice.

SQL, HTML and CSS are different types of language. While not considered “real” programming languages as such (on their own you could not write a software application) they are essential to our CMS.

SQL (*Sequential Query Language*) is the de facto standard language of databases. While most databases today use some form of SQL to extract, view and alter data, the “dialect” differs from database to database. We will use SQL to fetch our dynamic content from our database.

HTML (*Hyper Text Markup Language*) is the language of the web page. Each document has separate elements e.g. a body, header, images etc. and the HTML standard defines what these elements are. HTML pages are served by Apache and interpreted by the client browser e.g. Firefox.

CSS (*Cascading Style sheets*) are used in conjunction with HTML to change the style of the raw HTML pages. While it would be possible to write a CMS without it, it would probably not be very aesthetic.

JavaScript is a lightweight programming language used for dynamic tasks in conjunction with HTML e.g. changing content on the fly. It is run seamlessly from the client browser.

Generally, programming languages fall into 2 categories, compiled and interpreted. For instance C, Basic and Pascal are compiled whereas most script languages are interpreted. The major difference between compiled and interpreted languages is how the program itself is accessed and run. In the compiled scenario, the initial source code is passed through a compiler which generates a stand-alone binary if the source code is valid. The operating system then handles the corresponding output. A binary compiled for one particular Operating System will not run on another – in general the compiler has to match the O/S unless some form of emulation and library support is available. With interpreted languages each line of the source code is passed through the interpreter which handles the corresponding output. Both language types sup-

port additional libraries which extend the core functionality of the language (e.g. graph support) and these are used as required. See Figure 1 and Figure 2 – Compiled and Interpreted languages.

The bottom line is that you need to choose your language for the task you have in hand. Some all purpose languages are great but you need to remember the limitations. The author often uses PHP for add-hoc scripts, but Perl or Bash would be just as effective. Often it is a case of what you feel most comfortable with, but at the same time you don't want to fall into the trap "When the only tool you have is a hammer every problem is a nail".

To err is Human

Writing code is paradoxically both infinitely creative and flexible yet structured and pedantic. One missed semicolon, a full stop in the wrong place, even word case can be the difference between a working code segment and an esoteric error message. Sometimes by fixing one problem other problems are introduced, sometimes the real problem was never addressed at all. It is important that we are

able to snapshot and document our changes as well as quickly isolate any problems. As part of the series we will look at version control and debugging.

The Draft Specification

The initial specification of our CMS is per Table 1. Further additions may be made over the series to demonstrate specific principles. The inspiration for parts of the specification came from the excellent CMS, Drupal by Dries Buytaert.

Testing

It is critical that any application is properly tested before release. While automated testing methods are available, for the purpose of this series will limit testing to some crude load and security testing and ensuring that the program "just works as advertised".

The Development Environment

In a commercial environment, the bare minimum would probably consist of a test (development) server, a live

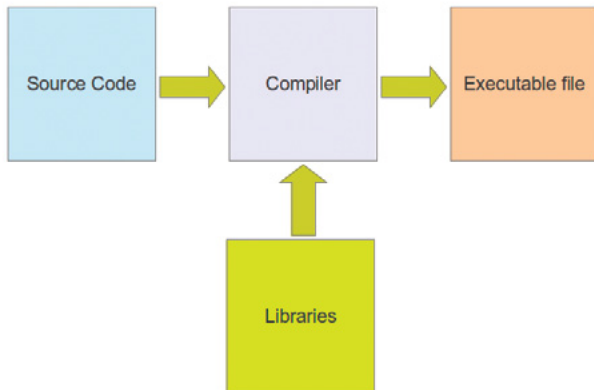


Figure 1. A compiled program

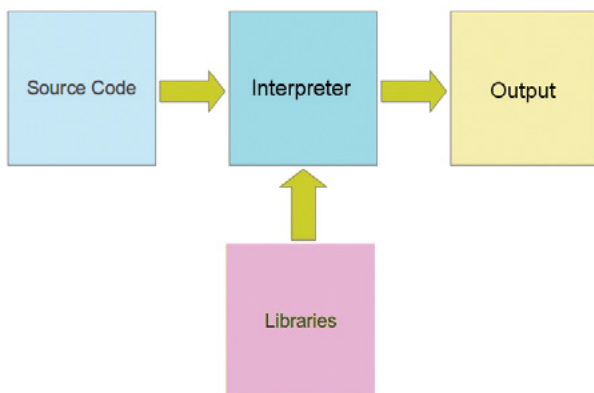


Figure 2. An interpreted script

Table 1. CMS draft specification

Initial CMS Specification
Allow an authorised user to create a W3C valid web page Database transactions (MySQL InnoDB storage engine) Efficient search Image and attachment uploads Menu module Modular and extensible Run under a standard AMP stack with little modification Support XHTML 1.0 strict Taxonomy Template and region driven – separate the rendering logic from page content Visitor statistics

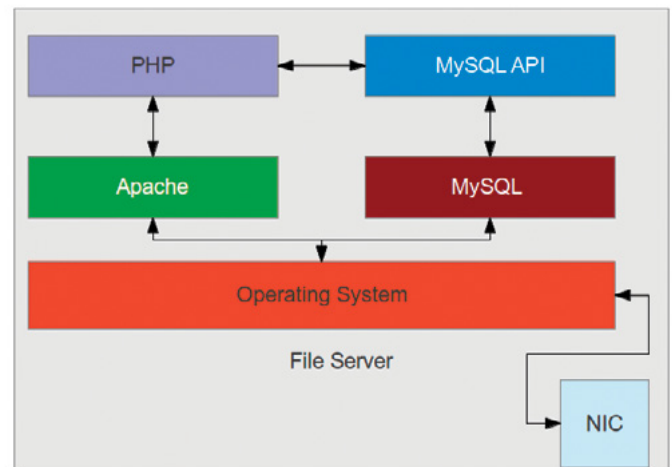


Figure 3. Our CMS architecture

(production) server, a Version Control Server (VCS), possibly a database server (MySQL) and the developers workstation with an Integrated Development Environment (IDE) for code development, syntax checking and debugging. Source code would be pulled from the VCS, edited and tested on either the workstation or the development server, committed to VCS and pushed to the production server for access by the users when stable and ready for release. This scenario is too complex for our series, but while it is possible to develop just from the command line, debugging (and certainly testing) will be close to impossible outside of a graphical environment. As a very bare minimum, you will need a headless Free-BSD box (without any GUI) and some sort of workstation with Firefox installed, but ideally your BSD development box should support Firefox, Netbeans, Apache, PHP, GIT and MySQL. Your favorite CLI editor can of course still be used for editing.

In the Next Article

We will start programming in earnest and start serving our first CMS page.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

The BSD Certification Group Inc. (BSDCG) is a non-profit organization committed to creating and maintaining a global certification standard for system administration on BSD based operating systems.

? WHAT CERTIFICATIONS ARE AVAILABLE?

BSDA: Entry-level certification suited for candidates with a general Unix background and at least six months of experience with BSD systems.

BDSP: Advanced certification for senior system administrators with at least three years of experience on BSD systems. Successful BDSP candidates are able to demonstrate strong to expert skills in BSD Unix system administration.

✓ WHERE CAN I GET CERTIFIED?

We're pleased to announce that after 7 months of negotiations and the work required to make the exam available in a computer based format, that the BSDA exam is now available at several hundred testing centers around the world. Paper based BSDA exams cost \$75 USD. Computer based BSDA exams cost \$150 USD. The price of the BDSP exams are yet to be determined.

Payments are made through our registration website:
<https://register.bsdcertification.org/register/payment>

i WHERE CAN I GET MORE INFORMATION?

More information and links to our mailing lists, LinkedIn groups, and Facebook group are available at our website:
<http://www.bsdcertification.org>

Registration for upcoming exam events is available at our registration website:
<https://register.bsdcertification.org/register/get-a-bsdcg-id>

FreeBSD Programming Primer – Part 2

In the second part of our series on programming, we will look at configuring our development server, write our first lines of code and commit the changes to a version control system.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

Before we get started, you need to have a FreeBSD test server available with the AMP (Apache / MySQL / PHP) installed. We will also use a version control system (VCS) and a CLI based text editor. I am using FreeBSD 9.0 with VI, MC (for file management) and GIT running under Virtualbox.

Start by installing FreeBSD from DVD and configure networking, user and root accounts, etc. as normal.

Key

- Command line instructions
- Alterations to configuration files
- MySQL prompt / SQL
- HTML / XHTML / PHP code

Part 1. Installing the Software

Step 1

As root, Install mc and git from packages:

```
dev# pkg_add -r mc git
```

Step 2. Upgrade the Ports Tree

```
dev# portsnap fetch && portsnap extract
```

Step 3. Install Apache

```
dev# cd /usr/ports/www/apache22
dev# make install clean
```

Configure `rc.conf` to start Apache on reboot:

```
dev# echo 'apache22_enable="YES"' >> /etc/rc.conf
```

Ensure hosts has your machine name set in `/etc/hosts` otherwise Apache will not start.

```
:::1      localhost dev
127.0.0.1      localhost dev
```

Start Apache:

```
dev# /usr/local/etc/rc.d/apache22 start
```

Step 4. Install MySQL

```
dev# cd /usr/ports/databases/mysql55-server
dev# make install clean
```

Start MySQL:

```
dev# echo 'mysql_enable="YES"' >> /etc/rc.conf
```



```
dev# /usr/local/etc/rc.d/mysql-server start
```

Set the MySQL root password and check MySQL works:

```
dev# /usr/local/bin/mysqladmin -u root password 'cms-
password'

dev# rehash
dev# mysql -uroot -pcms-password

mysql>\q
```

Step 5. Install PHP5 and Language Extensions

Enable and build apache module. See Figure 1.

```
dev# cd /usr/ports/lang/php5
dev# make config
```

Install PHP5 and the extensions:

```
dev# make install clean
```

Enable mysql and mysqli support. See Figure 2.

```
dev# cd /usr/ports/lang/php5-extensions/
dev# make config
dev# make install clean
```

Edit `/usr/local/etc/apache22/httpd.conf` to reflect the following:

```
DirectoryIndex index.html index.xhtml index.php
```

And add the following at the end for PHP support:

```
AddType application/x-httpd-php .php
AddType application/x-httpd-php-source .phps
```

Copy the `php.ini` file across:

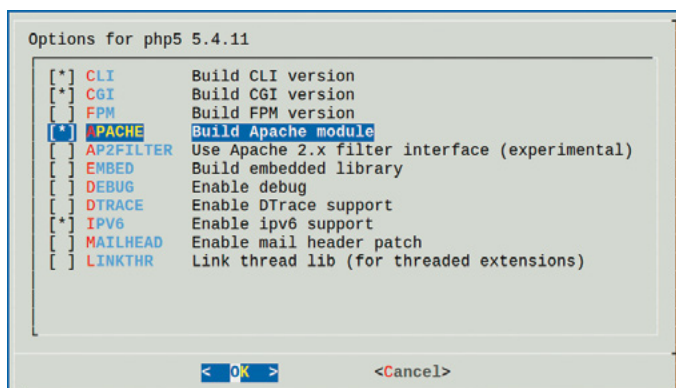


Figure 1. Enabling the Apache module

```
dev# cp /usr/local/etc/php.ini-development
/usr/local/etc/php.ini
```

Restart apache to pick up the new PHP extensions:

```
dev# /usr/local/etc/rc.d/apache22 restart
```

Now we need to setup a development area in our home directory. We will create an account with username dev:

```
dev# adduser
```

Follow the prompts (the defaults are fine), and give the new user a password. We want to edit `/develop` as dev, so move the apache data directory across to `/home/dev` and symlink back. That way, Apache can serve the files we create as a non-root user as we can run GIT as a normal user:

```
dev# mv /usr/local/www/apache22/data/ /home/dev/
dev# chown dev:dev datapwd
dev# ln -s /home/dev/data/ /usr/local/www/apache22/data
dev# cd /home/dev/data
dev# chown dev:dev index.html
dev# /usr/local/etc/rc.d/apache22 restart
```

If you visit your dev box with a browser (<http://yourip-address>) you should see the standard Apache “It works!” welcome page.

Part 2. GIT Revision Control and our Test Pages

As a developer, a version control system is an important tool not only to track code changes, but to allow quick recovery from mistakes. Once a file is added and committed to the repository, any errors can be quickly rectified by rolling back to a previous version.

Login with (or su to) the new DEV user account, change to the data directory, and create a new repository then

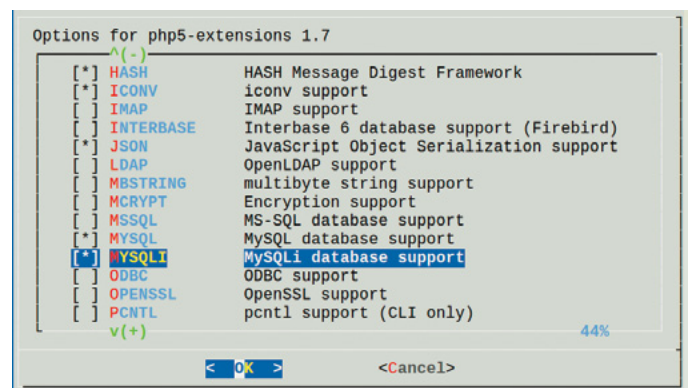


Figure 2. Enabling MySQL support

commit index.html to it after setting your details. When prompted in the editor, the commit message should be “Initial Load”.

```
dev# su dev
dev# cd /home/dev/data/
dev# git config --global user.name "dev"
dev# git config --global user.email dev@dev
dev# git init
dev# git add *
dev# git commit
```

PHP Version 5.4.11	
	
System	FreeBSD dev 9.0-RELEASE FreeBSD 9.0-RELEASE #0: Tue Jan 3 07:15:25 UTC 2012 root@obrian.cse.buffalo.edu:/usr/obj/usr/src/sys/GENERIC i386
Build Date	Feb 2 2013 00:08:03
Configure Command	'./configure' '--with-layout=GNU' '--localstatedir=/var' '--with-config-file-scan-dir=/usr/local/etc/php' '--disable-all' '--enable-libxml' '--enable-mysqld' '--with-libxml-dir=/usr/local' '--with-pcre-regex=/usr/local' '--with-zlib-dir=/usr' '--program-prefix=' '--with-apxs2=/usr/local/sbin/apxs' '--with-regex=php' '--with-zend-vm=CALL' '--prefix=/usr/local' '--mandir=/usr/local/man' '--infodir=/usr/local/info' '--build=i386-portbld-freebsd9.0'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/usr/local/etc
Loaded Configuration File	/usr/local/etc/php.ini
Scan this dir for additional .ini files	/usr/local/etc/php
Additional .ini files parsed	/usr/local/etc/php/extensions.ini
PHP API	20100412
PHP Extension	20100525
Zend Extension	220100525
Zend Extension Build	API220100525.NTS

Figure 3. PHP enabled

This will commit the original index.html to the new GIT repository. Edit index.html to reflect Code Listing 1 – “Hello World” is always the first statement written in experimental code. Check with your browser that the page has changed (you may need to press Shift F5 to refresh the cache). Now commit it to the repository:

```
dev# git commit -am "First line of HTML"
```

To view the change log:

```
dev# git log
```

Now delete index.html. To recover:

```
dev# git checkout index.html
```

```
$ git log
commit 30e37cee5475ded94d4eb6ba04c68b3b941af0cc
Author: dev <dev@dev>
Date: Sat Feb 2 03:12:58 2013 +0000

    ~XHTML and PHP test page

commit 0007073d6679e0e09397787d4c4abd1614d46e0f
Author: dev <dev@dev>
Date: Sat Feb 2 02:08:48 2013 +0000

    Initial load

$
```

Figure 4. Git log

Listing 1. The modified Apache index.xhtml

```
<html><body><h1>Hello World!</h1></body></html>
```

Listing 2. index.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>My first XHTML page</title>
</head>
<body>
  <p>Hello world</p>
</body>
</html>
```

Listing 3. phpinfo.php

```
<?php phpinfo();
```


Further reading

- GIT VCS – <http://githowto.com>
- PHP – <http://php.net>
- W3 Schools – <http://www.w3schools.com>
- W3C – <http://www.w3.org>

To go back to the original Apache file (Where 0007073d is the first 8 digits of the file checksum) and overwrite your changes permanently:

```
dev# git checkout 0007073d
```

Now the log will only show the original file. Create two files `index.xhtml` and `phpinfo.php` with the code from code Listing 2 and 3 respectively and add and commit to the repository:

```
dev# git add *
dev# git commit -am "XHTML and PHP test page "
dev# git log
```

You should see a log file similar to Figure 4.

Listing 1 is a standard XHTML page, with the XML and document type defined. In the next article, we will look at adding CSS and Javascript to this skeleton, but the important point to note here is that all the tags are “balanced” – every opening tag (e.g. `<p>`) has to have a matching closing tag. To view this page, visit <http://youripaddress/index.xhtml> in your browser.

Listing 2 is a very simple PHP command – `phpinfo()`; displays all the configuration values, modules loaded etc. available to the PHP interpreter. You should see a page similar to Figure 3 if you visit <http://youripaddress/phpinfo.php>.

In the Next Article

We will look at code structure, program flow and how to embed CSS and Javascript in our pages. We will also start using SQL to dynamically generate pages.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

If you wish to contribute to BSD magazine, share your knowledge and skills with other BSD users – do not hesitate – read the guidelines on our website and email us your idea for an article.

Join our team!



Become BSD magazine Author or Betatester

As a betatester you can decide on the contents and the form of our quarterly. It can be you who read the articles before everybody else and suggest the changes to the author.

Contact us:
editors@bsdmag.org
www.bsdmag.org

FreeBSD Programming Primer – Part 3

In the third part of our series on programming, we will look at code structure, program flow and how to embed CSS and Javascript in our pages. We will also start using SQL to dynamically generate web pages.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

Before we start coding in earnest, we will look at the basic construction of our programming language (PHP), the directory and functional structure of our CMS and how this all fits together.

Our CMS will be designed to be as extensible as possible, and will follow the design as detailed in Figure 1. Pages will be stored in the MySQL database, merged with the header, templates, CSS and Javascript and returned to the client browser. This will allow us to separate design from content efficiently.

Part 1. PHP Fundamentals

Any language – both verbal and programming – comprises of separate elements that fit together in a logical structure that communicates meaning to the recipient. For language to be effective, rules are strictly defined, not only to preserve efficiency but to prevent misunderstanding. For example, a written sentence will comprise of nouns, verbs and adjectives – likewise computer code will consist of variables, expressions and control structures. The main functional difference between a human language such as English and a programming language is flexibility and interpretation – as humans we are adaptable enough to interpret a missing full stop or misspelled word correctly, whereas a computer will fail miserably.

Here we will look at some of the basic building blocks of PHP.

The following code examples can be created in the examples directory using your favorite editor (in this case I am using VI). Login to the webserver using SSH or from the console, switch to the DEV account from root and create the files:

```
dev# su
dev# su dev
dev# cd /home/dev/data/
dev# mkdir examples
```

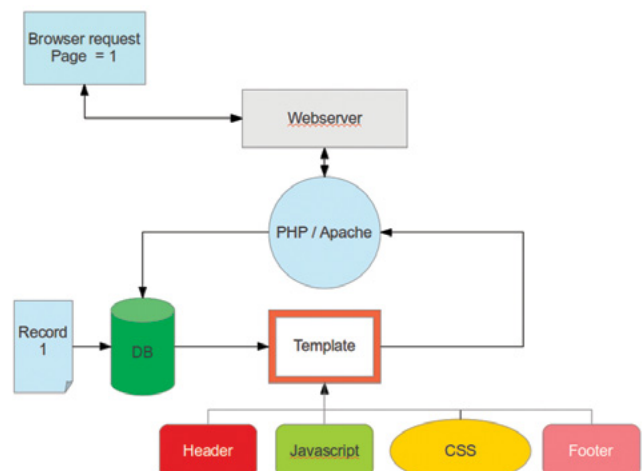


Figure 1. CMS design

```
dev# cd examples
dev# vi example1.php
```

The example can then be run from: <http://yourserveripaddress/examples/example1.php> (see Figure 2).

You do not need to run the examples to develop a working CMS, but they are useful to experiment with. Change values and experiment.

PHP tags

All PHP code requires an opening `<?php` tag. The closing `?>` tag is only necessary when combining PHP with Javascript, HTML etc.

Comments

In PHP comments are denoted with `//`. A block of comments can also be with a block using `/* ... */`.

Expressions

To quote the PHP website “Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is “anything that has a value”.

For instance, if the server has 3 disk drives this could be written as:

```
<?php
$disk_drives = 3;
```

Constants

A constant is useful where the the value will not change across the execution of the script. Unlike variables, constants are accessible within function calls and throughout the script, whereas variables may be local only to that particular function. This aids program readability.



Figure 2. *example1.php* running via a browser

The circumference of a circle is calculated by multiplying the diameter by PI (3.14). As PI is a constant and will not change, we can define PI as a constant. See Listing 1.

Variables

A variable holds the result of an expression or statement. As the name implies, the value of the variable will change over the life of the program. The variable name is defined on the left hand side of the `=` sign and the value of the variable is defined on the right hand side see Listing 2.

Data types

Data types, as the name suggests, define what type of data we can hold in a variable or constant. The basic types are booleans, integers, floats, strings, and arrays.

Booleans

A boolean is used where a dual state is useful. A boolean has one of two values, TRUE or FALSE. For instance, we can define the constant DEBUG and act accordingly depending on how DEBUG evaluated by the “if” control structure: see Listing 3.

Listing 1. *example1.php*

```
<?php

/*
 * example1.php
 * Constants
 * Define PI as the constant 3.14 and output the result.
 *
 */

define("PI", 3.14);
echo PI;
```

Listing 2. *example2.php*

```
<?php

/*
 * example2.php
 * Variables
 * Define circumference as the variable $circumference
 * with the value
 * 12.775 and output the result
 *
 */

$circumference = 12.775;
echo $circumference;
```


Integers

An integer is a whole number of the set $\mathbb{Z} = \{\dots, -2, -1, 0, 1, 2, \dots\}$. As the maximum and minimum value is platform independent, we can access this value via the constant `PHP_INT_MAX`. If PHP encounters a number larger than `PHP_INT_MAX`, the number will be converted to a float. See Listing 4.

Floats

The maximum size of a floating point number, like Integers, is platform dependent. However, all sorts of rounding and logic errors can be introduced into code with floats as they behave differently from integers so particular care should be used. For instance, $1/3$ is equal to 0.33333 with 3 recurring to infinity, but as we have limited space it is impossible to represent this fully. If accuracy is critical, various libraries are available to improve float functionality. See Listing 5.

Strings

A PHP string can contain up to 2Gb of characters. A string is limited to 256 ASCII characters, and does not internally store strings in Unicode format unlike some other languages.

A string can be surrounded either by single or double quotes. If surrounded by double quotes, PHP will

endeavor to evaluate any variables contained within. A single quoted string is called a string literal as the string is interpreted literally rather than expanding any variables contained within it. Like the Vi versus Emacs discussion, the use of single or double quotes is very much a question of what you want to achieve. While single quotes may be considered quicker than double quotes, other coding factors have a greater impact on speed. See Listing 6.

Arrays

An array is a list with a key and value pair. For instance, we can list the major BSD distributions as an array variable, rather than multiple separate variables. We can then perform operations on the list by looping through the key/value pairs.

Arrays are useful where we have to keep records in sync. For instance if the records from a database table were dumped into an array it would be easy to manage using the record ID as key. If separate variables were used, it would be difficult to manage and the potential for errors would be great. Arrays do not need to have sequential keys, indeed PHP supports the mixing of numeric and string values as keys within arrays. You can even define another array as the value of an array

Listing 3. *example3.php*

```
<?php

/*
 * example3.php
 * Booleans
 * Define DEBUG as a boolean, and print our status.
 * Change TRUE to FALSE to change the output message.
 *
 */

define("DEBUG", TRUE);
if (DEBUG) {
    echo 'We are in Debug mode';
} else {
    echo 'We are not in Debug mode';
}
```

Listing 4. *example4.php*

```
<?php

/*
 * example4.php
```

```
 * Integers
 * Check the maximum integer size available on your
   platform.
 * For a 32 bit system this will be 2147483647.
 *
 */

echo PHP_INT_MAX;
```

Listing 5. *example5.php*

```
<?php

/*
 * example5.php
 * Floats
 * Calculate PI as a float using the more accurate
   formula 22 / 7.
 * This should return 3.1428571428571.
 *
 */

$pi = 22 / 7;
echo $pi;
```

The first array example is the traditional PHP method for defining arrays, the second version onwards is available in > PHP 5.4. See Listing 7.

Operators

Operators are used to compare values (Comparison / Logical) or change values (Arithmetic / Assignment / Increment / Decrement / String). Care has to be taken with operator precedence, as certain operators will fire first. For example, the multiplication operator `*` takes precedence over the addition operator `+` unless the addition portion of the equation is wrapped in brackets: see Listing 8. See Table 1 for the full list of the most common operators.

Functions

Functions are small blocks of code that can act as effectively as a “black box” to the code that is calling it. As functions can be called repeatedly from anywhere within code – and if written properly – will provide a consistent result. Functions can act independently of the code that is calling them, or can return a result that can be manipulated by the main body of the program. An important point to realize is that variables defined inside a function are generally out of scope of the main body – that is to say `$a` in the main body of a program cannot be accessed by the function unless it is either passed as a parameter or accessed via some other method (PHP has a rich library of internal functions, if

Listing 6. *example6.php*

```
<?php

/*
 * example6.php
 * Strings
 * Demonstration of the PHP string type.
 * Note that the last line is functionally identical to
   the previous
 * line and we are separating each line with a HTML <br />.
 *
 */

define("BR", '<br />');
$pi = 22 / 7;
echo 'The value of PI is: $pi' . BR;
echo 'The value of PI is: ' . $pi . BR;
echo "The value of PI is: $pi" . BR;
```

Listing 7. *example7.php*

```
<?php

/*
 * example7.php
 * Arrays
 * All of these examples are functionally equivalent
 *
 */

define("BR", '<br />');

// Define the array then print it out using the function
print_r()
// Use BR to separate each line
```

```
$array_1 = array(
    "0" => "FreeBSD",
    "1" => "OpenBSD",
    "2" => "NetBSD",
);

print_r($array_1);
echo BR;

$array_2 = [
    "0" => "FreeBSD",
    "1" => "OpenBSD",
    "2" => "NetBSD",
];

print_r($array_2);
echo BR;

// Arrays can use mixed key values - they do not have to start at 0

$array_3[5] = "FreeBSD";
$array_3["This is key 6"] = "OpenBSD";
$array_3[7] = "NetBSD";

print_r($array_3);
echo BR;

// Let PHP assign the key values

$array_4[] = "FreeBSD";
$array_4[] = "OpenBSD";
$array_4[] = "NetBSD";

print_r($array_4);
echo BR;
```

you do not recognize a function call in the later CMS sample code the script will be using a built in PHP function. The same applies to the javascript sample). See Listing 9.

Control structures

Control structures, along with Comparison / Logical operators provide the logic for our program. Example 3 is a good example of the if/else control structure.

These are only a very small subset and the most common of the extensive features available with PHP. To see the full list, please visit the PHP language guide at <http://www.php.net/manual/en/langref.php>.

Part 2. CMS Structure

See Table 2 – CMS directory structure. Create the directories and the 14 files as per the instructions for the example code under Part 1 – PHP fundamentals. Before we can start coding in earnest, we need to populate our MySQL database.

Create the following files, and create the database, table and our first page stored in the database: see Listings 10-12. Note that the Ipsum Lorem test should be on one line with no carriage returns or line feeds. Your editor may wrap this very long line. Create the the database, table and page as follows in Listing 13.

Table 1. PHP Operators

Example	Name	Result	Operator
-\$a	Negation	Opposite of \$a.	Arithmetic
\$a + \$b	Addition	Sum of \$a and \$b	
\$a - \$b	Subtraction	Difference of \$a and \$b	
\$a * \$b	Multiplication	Product of \$a and \$b	
\$a / \$b	Division	Quotient of \$a and \$b	
\$a % \$b	Modulus	Remainder of \$a / \$b	
\$a = 3	Assignment	Sets \$a to 3	Assignment
\$a += 5	Assignment	Sets \$a to 8	
\$a = 'Hello '	Assignment	Sets \$a to 'Hello'	
\$a .= 'world'	Assignment	Sets \$a to 'Hello world'	
\$a == \$b	Equal	TRUE if \$a is equal to \$b after type juggling.	Comparison
\$a === \$b	Identical	TRUE if \$a is equal to \$b, and they are of the same type.	
\$a != \$b	Not equal	TRUE if \$a is not equal to \$b after type juggling.	
\$a <> \$b	Not equal	TRUE if \$a is not equal to \$b after type juggling.	
\$a !== \$b	Not identical	TRUE if \$a is not equal to \$b, or they are not of the same type.	
\$a < \$b	Less than	TRUE if \$a is strictly less than \$b.	
\$a > \$b	Greater than	TRUE if \$a is strictly greater than \$b.	
\$a <= \$b	Less than or equal to	TRUE if \$a is less than or equal to \$b.	
\$a >= \$b	Greater than or equal to	TRUE if \$a is greater than or equal to \$b.	
\$a and \$b	And	TRUE if both \$a and \$b are TRUE.	Logical
\$a or \$b	Or	TRUE if either \$a or \$b is TRUE.	
\$a xor \$b	Xor	TRUE if either \$a or \$b is TRUE, but not both	
! \$a	Not	TRUE if \$a is not TRUE.	
\$a && \$b	And	TRUE if both \$a and \$b are TRUE.	
\$a \$b	Or	TRUE if both \$a and \$b are TRUE.	
++\$a	Pre-increment	Increments \$a by one, then returns \$a.	Inc / Dec
\$a++	Post-increment	Returns \$a, then increments \$a by one.	
--\$a	Pre-decrement	Decrements \$a by one, then returns \$a.	
\$a--	Post-decrement	Returns \$a, then decrements \$a by one.	

Listing 8. example8.php

```
<?php

/*
 * example8.php
 * Demonstration of operator precedence
 *
 */

define("BR", '<br />');

$a = 1 + 5 * 3;
$b = (1 + 5) * 3;

echo '$a will evaluate to 16: ' . $a . BR;
echo '$b will evaluate to 18: ' . $b . BR;
```

Listing 9. example9.php

```
<?php

/*
 * example9.php
 * Demonstration of a function call
 *
 */

// As BR is a constant, this is available to our function directly

define("BR", '<br />');

// $pi is not available to our function, we will need to
// access it by
// other methods

$pi = 22 / 7;

echo "Circumference with a diameter 5: " . print_circ1(5);
echo "Circumference with a diameter 10: " . print_circ2(10,$pi);

function print_circ1($diameter) {

    // print_circ1() will display $pi * $diameter
    // Define $pi as a global variable

    global $pi;

    // As BR is a global constant we can access it directly
    // Return our result to the main body of the program
```

```
    return $pi * $diameter . BR;
}

function print_circ2($diameter, $pi) {

    // print_circ2() will display $pi * $diameter
    // As BR is a global constant and $pi has been passed to our
    // function we can access them directly.
    // Return our result to the main body of the program

    return $pi * $diameter . BR;
}
```

Listing 10. createdb.sql

```
create database freebsdcm;
grant usage on *.* to bsduser@localhost identified by
    'cmsdbpassword';
grant all privileges on freebsdcm.* to bsduser@localhost;
```

Listing 11. createpagetbl.sql

```
CREATE TABLE if not exists pages (
    id INT NOT NULL AUTO_INCREMENT,
    PRIMARY KEY(id),
    title VARCHAR(50) NOT NULL,
    h1 VARCHAR(50),
    body TEXT
);
```

Listing 12. createpage.sql

```
USE freebsdcm;
INSERT INTO pages
VALUES (
    '',
    'My first page',
    'Page header',
    'Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris
    interdum auctor tellus sed dignissim. Phasellus non orci massa,
    nec feugiat sem. Vestibulum molestie interdum
    bibendum. Nunc quis elit nulla, sit amet rutrum lorem.
    Quisque
    odio est, sagittis nec accumsan ut, placerat sit
    amet lectus. Curabitur aliquam dignissim felis, a malesuada leo
    fringilla at. Sed ornare aliquet lacus, quis
    imperdiet augue mattis eu. Nulla porta odio ut erat
    consectetur at
    molestie justo suscipit. Aenean convallis
    pellentesque nisl, vitae posuere mauris facilisis vitae.
    Morbi in
    tellus nisl, vel facilisis diam.'
);
```

Part 3. The Code

The following PHP files contain the code for our website. Create each file as per Table 2. See Listing 14.

global.css holds the style information for our site. Experiment with the font sizes, line spacing etc. to style the site to your liking. If you use Firefox, install the Firebug plugin to dynamically change the values, but if you want to make them permanent you will need to edit this file. Also try



Figure 3. Our first page – index.php

renaming global.css and refreshing your browser's cache to see the effect of the styling on the site. See Listing 19.

The include files build our basic HTML header and footers, add the CSS via global.css and load the Javascript. See Listing 20-22. The javascript files

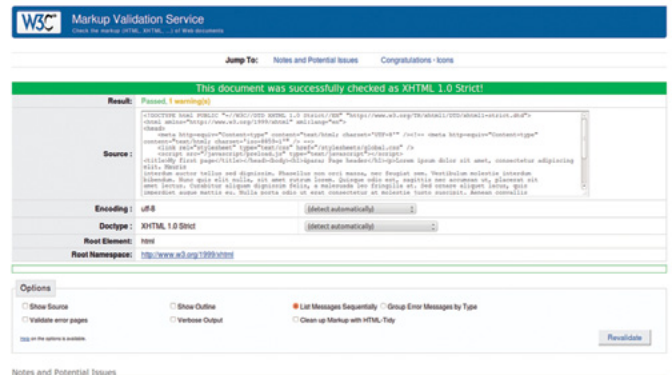


Figure 4. The page validates

Listing 13. Creating the database, table and pages in MySQL

```
#dev mysql -u root password 'cms-password' < createdb.sql
#dev mysql -u root password 'cms-password' < createpagetbl.sql
#dev mysql -u root password 'cms-password' < createpage.sql
```

Table 2. CMS directory structure

CMS Directory structure		
All directories are under /usr/local/www/apache22/data		
Directory / file	Purpose	Files
examples	Example PHP code	example1.php example2.php example3.php example4.php example5.php example6.php example7.php example8.php example9.php
includes	PHP includes for CMS. Each file contains specific functionality as named	cms.inc core.inc html.inc mysql.inc
index.php	Start page for our CMS	index.php
javascript	Javascript support for our website	postload.js preload.js
sql	Contains the SQL loader scripts for our website.	createdb.sql createpage.sql createpagetbl.sql
stylesheets	Holds the CSS stylesheets for the website	global.css
templates	Holds the templates for the website	header.inc footer.inc template.inc

Listing 14. index.php

```
<?php

/*
 * index.php
 * Index page for FreeBSD CMS
 *
 */

// Get required files

// Our global settings - Note need full path
require_once 'includes/cms.inc';
// Core functions
require_once INCLUDES.'core.inc';

// HTML functions
require_once INCLUDES.'html.inc';

// MySQL functions
require_once INCLUDES.'mysql.inc';

// Turn full debug functionality on if enabled

if(DEBUG){

    // Turn on full PHP error reporting
    error_reporting(E_ALL);

}else{

    // Hide all error messages
    error_reporting(0);

}

// Build page - use first record in database

$page['id'] = 1;

buildpage($page);
```

Listing 15. cms.inc

```
<?php

/*
 *
 * cms.inc
```

```

 * Contains default settings for our CMS
 *
 * NOTE: ¶ denotes a line wrapped - all code should be
 *       on one line
 *
 */

// Set our timezone

date_default_timezone_set('Europe/London');

// Copyright details

define("LICENCE", 'licence.txt');
define("COPYRIGHT", 'Copyright &copy; 2013 Rob Somerville
    ¶ me@merville.co.uk');
define("COPYYEAR", date('Y'));
define("COPYAUTH", 'Rob Somerville');
define("COPYEMAIL", 'me@merville.co.uk');

// Version

define("VERSION", 'Version 1.0 not for production use');

// Mode - If DEBUG is set to true, show errors and debug
info

define("DEBUG", TRUE);

// Where to find our files

define("TEMPLATES", 'templates/');
define("INCLUDES", 'includes/');
define("SQL", 'sql/');

// HTML tags that are orphaned and not defined in out
template files

define("BODY", '<body>');
define("HEAD", '</head>');

// MySQL details

define("DBSERVER", 'localhost');
define("DBUSER", 'bsduser');
define("DBPASSWORD", 'cmsdbpassword');
define("CMSDB", 'freebsdcms');
```


Listing 16. core.inc

```

<?php

/*
 *
 * core.inc
 * Contains core functions for our CMS
 *
 */

function buildpage($page) {

    // Builds a standard page

    $id = $page['id'];

    // Build the SQL and get the result

    $sql = "SELECT * FROM pages WHERE id='$id' LIMIT 1";
    $result = mysql_select($sql);

    // Output our page header

    outfile(TEMPLATES . 'header.inc');

    // Create our body

    $markup = '';
    $markup .= wraptag('title', $result[4]);
    $markup .= HEAD;
    $markup .= BODY;

    // If we are in debug mode, show an alert

    if(DEBUG){

        $debug = '&para; ';

    }else{

        $debug = '';

    }

    // Add to markup

    $markup .= wraptag('h1',$debug . $result[3]);
    $markup .= wraptag('p',$result[5]);
    $markup .= divclass(ahref(COPYRIGHT, LICENCE,
'Copyright and

```

```

    licence details'),'licence');

    // Output all our markup

    echo $markup;

    // Output our HTML page footer

    outfile(TEMPLATES . 'footer.inc');

}

function outfile($file) {

    // Outputs template file to browser e.g header,
    footer, license etc.

    $fh = fopen($file, 'r');

    while (!feof($fh)) {
        echo fgets($fh);
    }

    fclose($fh);
}

```

Listing 17a. html.inc

```

<?php

/*
 *
 * html.inc
 * Contains core html functions for our CMS
 *
 */

function wraptag($tag, $text) {

    // Wraps $text with compliant tags
    // wraptag('p',sometext)
    // <p>sometext</p>

    return '<' . $tag . '>' . $text . '</' . $tag . '>';
}

function divclass($divcontent, $class, $id = '') {

    // Generates a div tag $text with compliant tags
    // divclass('content','class')

```

Listing 17b. html.inc

```
// <div class="class">content</div>
// divclass('content','class','id')
// <div class="licence" id="id">content</div>

if ($id != '') {

    $id = 'id="' . $id . '"';
}

return '<div class="' . $class . '" ' . $id . '>' .
    $divcontent . '</div>';
}

function ahref($text, $url, $title = '') {

    // Generates an href tag $text with compliant tags
    // ahref('Click here',freebsd.org)
    // <a href="http://freebsd.org" title="Click
here">Click here</a>
    // ahref('Click here',freebsd.org,'Link title')
    // <a href="http://freebsd.org" title="Link
title">Click here</a>

    if ($title == '') {
        $title = $text;
    }

    $ahref = '<a href="' . $url . '" title="' . $title .
    '">' . $
    $text . '</a>';

    return $ahref;
}
```

Listing 18. mysql.inc

```
<?php

/*
 *
 * mysql.inc
 * Contains MySQL functions for our CMS
 *
 */

function mysql_select($sql) {

    $db = new mysqli(DBSERVER, DBUSER, DBPASSWORD, CMSDB);
```

```
if($db->connect_errno > 0){
    die('Unable to connect to database [' .
    $db->connect_error . ']');
}

if(!$result = $db->query($sql)){
    if(DEBUG){
        die('There was an error running the query [' .
        $db->error .
        ']');
    }else{
        die('');
    }
}

// Pass our results to an array to be returned

$r = array();

$r[] = $result->num_rows; // No of rows returned
$r[] = $db->affected_rows; // No of rows affected
                        e.g. 1
                        update /delete

while($row = $result->fetch_assoc()){
    $r[] = $row['id'];
    $r[] = $row['hl'];
    $r[] = $row['title'];
    $r[] = $row['body'];
}

// Free the result

$result->free();

return $r;
}
```

Listing 19. global.css

```

/* global.css - the site global stylesheet */

h1 {
    background-color: teal;
    float: left;
    color: white;
    padding: 21px;
    text-transform: uppercase;
}

p {
    float: left;
}

body {
    line-height: 160%;
    text-align: justify;
    float: left;
}

html {
    background: none repeat scroll 0 0 #F9F8F2;
    border: 1px solid;
    color: teal;
    font-family: Verdana;
    margin: 10px;
    padding: 20px;
}

.jstime, .licence {
    background: none repeat scroll 0 0 #EDEAC6;
    border: 1px solid #DADADA;
    color: slategrey;
    float: right;
    font-family: Verdana;
    font-size: x-small;
    margin-bottom: 5px;
    margin-right: 10px;
    margin-top: 10px;
    padding: 3px 10px;
}

```

Listing 20. header.inc

```

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//
EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
    <meta http-equiv="Content-type" content="text/html;
charset='iso- 8859-1'" />
    <link rel="stylesheet" type="text/css"
href="/stylesheets/global.css" />
    <script src="/javascript/preload.js"

```

```

type="text/javascript"></script>

```

Listing 21. footer.inc

```

<script src="/javascript/postload.js" type="text/"
javascript"></script></body></html>

```

Listing 22. template.inc

This can be empty, but needs to be created.

```

<!-- Template file -->

```

Listing 23. preload.js

```

/*
    preload.js
    Provides Javascript support
*/

// Call the function displaydate()

displaydate()

function displaydate(){

    // Displays the date and time in AM/PM format.

    var currentTime = new Date()
    var hours = currentTime.getHours()
    var minutes = currentTime.getMinutes()
    var month = currentTime.getMonth() + 1
    var day = currentTime.getDate()
    var year = currentTime.getFullYear()

    if (minutes < 10){
        minutes = "0" + minutes
    }

    var ampm = "";

    if(hours > 11){
        ampm = "PM"
    } else {
        ampm = "AM"
    }

    document.write("<div class=\"jstime\">" + day + "/" +
month + "/" +
year + " " + hours + ":" + minutes + ampm + "</div>")

```


Listing 24. postload.js

```
/*
  postload.js
  Just an empty file with comments
*/
```

Useful links

- W3C Validator (by file upload) – <http://validator.w3.org>
- PHP documentation – <http://www.php.net/manual/en>
- W3 Schools – <http://www.w3schools.com>

are split into 2. Preload.js provides the date and time on each page, postload.js is just an empty file which provides hooks we will use later on in the series. See Listing 23-24.

Part 4. Our Simple CMS

Once you have entered the code as per table 2, point your browser at <http://yourserveripaddress/index.php>. You should see a page similar to Figure 3. Turn debug off and on in cms.inc, and the paragraph mark should disappear. If you copy the HTML source from the page (In you browser view source, select all, copy and paste into the W3C validator) the page should validate.

So what is our code doing?

The unformatted text is stored as plain text in our database table. Index.php forms the first page of our website, and loads our settings and functions from the include files. The first stage is to load our header from a plain text file, which is the HTML at the start of our page. The header file in turn loads the CSS and javascript, and returns control to index.php. We then query the database, wrap the text in the relevant HTML tags and output to our browser. We then close the HTML with our footer HTML. In the next part of our series we will develop the CMS further, passing parameters via our browser to load different pages, We will also start using our template file so that we can design our site the way we want it with separate blocks and regions.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

a d v e r t i s e m e n t

IT-Securityguard

Lets secure IT



Android Vulnerability Scan



Web Penetration testing



Secure hosting

contact: contact@it-securityguard.com

www.it-securityguard.com

FreeBSD Programming Primer – Part 4

In the fourth part of our series on programming, we will continue to develop our CMS. Here we will examine how a modern CMS dynamically generates and controls content and implement a similar model in our PHP code.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP, and SQL code

What you should know...

- BSD and general PC administration skills

In the early days of the World Wide Web, HTML pages were literally handcrafted masterpieces of content. Before applications such as Dreamweaver arrived that allowed content providers to design attractive pages with the ease of a document produced in a word processor, it was a matter of writing copious amounts of HTML for each page, checking that the links and the HTML were correct, and repeating for each page. This model was highly inefficient, as not only was a lot of the HTML repeated across pages, the chances of errors coming in and either causing the page to render incorrectly or pointing to the wrong address became greater as the site grew. Managing a website with 100 pages is possible; a website with 10,000 pages a nightmare.

The complex sites we see today on the Internet would be impossible without the Content Management System. Yet even now, large innovative sites are moving away from the CMS model toward frameworks that consider the locally provided content to be only a part of the website with 3rd party content supplying a significant proportion of the content.

While the technology meets the ethos of the web in that data can be shared freely, it poses the web designer and brand manager with a huge challenge – how can we take disparate pieces of content and serve these in a “wrapper” that to our website visitors appears as if it seamlessly represents our brand values? How can we

divorce the business process from the presentation? Is it possible for a website to develop a unique “personality” while at the same time remaining fresh, dynamic and easily changeable?

These hurdles are being overcome with the use of CSS (Cascading Style Sheets) and templating technologies. While the CSS manages the color, fonts, size, etc. of the content, templates allow us to adjust the order and visibility of the content. For example, we want to generate widely different content (both from a stylized and literal

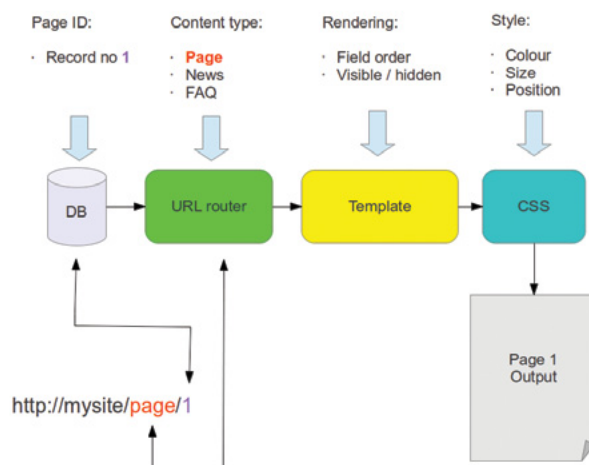


Figure 1. Page generation process

Great Specials

On FreeBSD® & PC-BSD® Merchandise

Give us a call & ask about our
SOFTWARE BUNDLES

1.925.240.6652

\$39.95

FreeBSD 9.1 Jewel Case CD Set
or FreeBSD 9.1 DVD

\$29.95

PC-BSD 9.1 DVD

\$49.95

The PC-BSD 9.0 Users Handbook
PC-BSD 9.1 DVD

\$99.95

The FreeBSD CD **or** DVD Bundle

Inside each CD/DVD Bundle, you'll find:
FreeBSD Handbook, 3rd Edition
Users Guide FreeBSD Handbook, 3rd Edition, Admin Guide
FreeBSD 9.1 CD or DVD set
FreeBSD Toolkit DVD



Stylish Dress Attire
Look Your Professional Best



Comfy Apparel
Stay Warm in Zip Ups & Pullovers

T-Shirts
Lots of Styles to Choose From

FreeBSD 9.1 Jewel Case CD/DVD \$39.95

CD Set Contains:

- Disc 1** Installation Boot LiveCD (i386)
- Disc 2** Essential Packages Xorg (i386)
- Disc 3** Essential Packages, GNOME2 (i386)
- Disc 4** Essential Packages (i386)

FreeBSD 9.0 CD \$39.95

FreeBSD 9.0 DVD \$39.95

FreeBSD Subscriptions

Save time and \$\$\$ by subscribing to regular updates of FreeBSD

FreeBSD Subscription, start with CD 9.1 \$29.95

FreeBSD Subscription, start with DVD 9.1 \$29.95

FreeBSD Subscription, start with CD 9.0 \$29.95

FreeBSD Subscription, start with DVD 9.0 \$29.95

PC-BSD 9.1 DVD (Isotope Edition)

PC-BSD 9.1 DVD \$29.95

PC-BSD Subscription \$19.95

The FreeBSD Handbook

The FreeBSD Handbook, Volume 1 (User Guide) \$39.95

The FreeBSD Handbook, Volume 2 (Admin Guide) \$39.95

The FreeBSD Handbook Specials

The FreeBSD Handbook, Volume 2 (Both Volumes) \$59.95

The FreeBSD Handbook, Both Volumes & FreeBSD 9.1 \$79.95

PC-BSD 9.0 Users Handbook \$24.95

BSD Magazine \$11.99

The FreeBSD Toolkit DVD \$39.95

FreeBSD Mousepad \$10.00

FreeBSD & PCBSD Caps \$20.00

BSD Daemon Horns \$2.00



Bundle Specials!
Save \$\$\$

Just Plain Fun
Mousepads & Novelty Horns



BSD Magazine
Available Monthly



For even **MORE** items
visit our website today!

www.FreeBSDMall.com

content perspective) depending on website section, page number and content type. See Figure 1 – Page generation process.

MySQL Interface

As it is important that we can quickly test our CMS, for those that would prefer the “Cut, Paste and Click” approach rather than managing long SQL statements via the command line, you can use a lightweight web-based database manager. The lightest of these (a single PHP page) is Adminer. An alternative is SQL buddy, and either of these can be quickly installed if desired by downloading the archive and extracting into a folder under the `/usr/home/dev/data`. The web-based interface can then be accessed from: <http://myserver/dirname>. See Table 1 – Useful links.

Adding New Content Types

At the moment, we only have one content type – a page. This is stored in the pages table and holds the following content as shown in Table 1.

Table 1. Page content from MySQL pages table

id	title	h1	body
1	My first page	Page header	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Mauris interdum auctor tellus sed dignissi...

This results in the following output as seen in Figure 2. Now let us create a second page in our database:

Method 1 – Via CLI

```
$ mysql -uroot -p'cms-password';

mysql> use freebsdcms;
mysql> INSERT INTO `pages` (`title`, `h1`, `body`)
-> VALUES ('My second page', 'H1', '2');
```

Method 2 – Via saved SQL statement

If you prefer, create a SQL file *createpage2.sql* in the SQL directory with the following content:

```
USE freebsdcms;
INSERT INTO `pages` (`title`, `h1`, `body`)
VALUES ('My second page', 'H1', '2');
```

Then execute this at the command line:

```
$ mysql -uroot -p'cms-password' < createpage2.sql
```

Method 3 – Via Adminer / SQL Buddy

Alternatively use the SQL command function in Adminer to execute the following SQL statement:

```
INSERT INTO `pages` (`title`, `h1`, `body`)
VALUES ('My second page', 'H1', '2');
```

Houston, We Have a Problem

We now have two pages in our database, but index.php still contains the following code:

```
// Build page - use first record in database
$page['id'] = 1;
buildpage($page);
```

This hard-wires index.php to only serve a page with an ID of 1. Depending on the URL passed to the webserver, we want to serve that type of content. For example *http://mysite/pages/1* will serve a page with an ID of 1, whereas *http://mysite/faqs/1* will serve an FAQ with an ID of 1, etc. Visiting *http://mysite* will return the home page (Page 1). This leads us to the next problem – where do we store the content types? We could include this in a separate MySQL table, but this would require an additional SQL query to be executed every time a page is loaded. As content types will not be changed very often, we can create another include file that defines our content

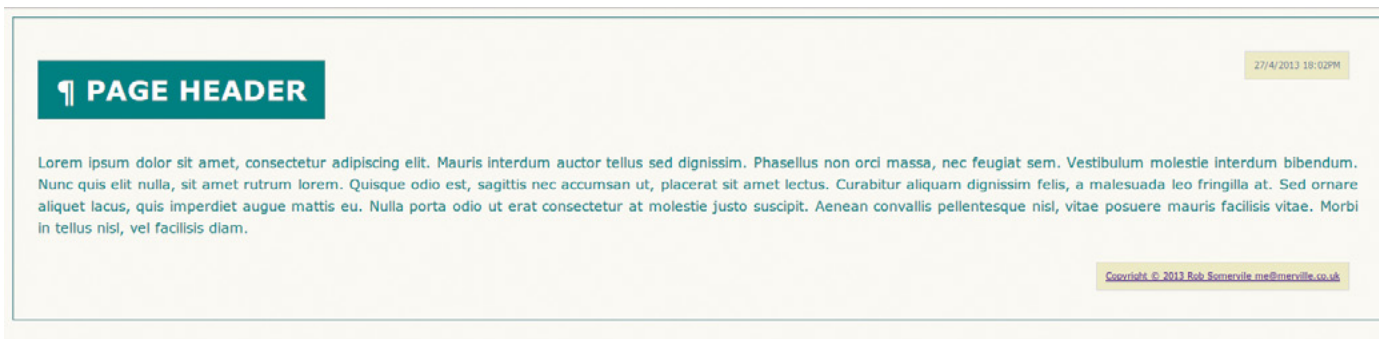


Figure 2. Our first page

Listing 1. *content.inc*

```
<?php
/*
 *
 * content.inc
 * Defines content types for our CMS
 *
 */
// Define the content type. This must match any tables
// defined in our
// CMS
$content_types[] = 'page';
$content_types[] = 'faq';
$content_types[] = 'news';
// Map each content type to a table. Each content type
// must be matched
// to a corresponding table
$content_tables['page'] = 'pages';
$content_tables['faq'] = 'faqs';
$content_tables['news'] = 'news';
```

Listing 2. *pages_template.inc*

```
<?php
/*
 *
 * pages_template.inc
 * Template for our page content type
 *
 * For content type foo the corresponding template would be:
 * foo_template.inc
 *
 * To display a field:
 * render($theme['name_of_field_as_defined_in_db']);
 *
 * To hide a field omit it from here
 * To change the rendering order, just re-order the fields
 *
 * NOTE: Any content generated by javascript will not be
 * managed here
 * A closing ?> tag is mandatory
 *
 */
render($theme['title']);
render($theme['debug']);
render($theme['h1']);
render($theme['timestamp']);
render($theme['body']);
render($theme['licence']);
?>
```

Listing 3. *index.php* replacement code

```
// First we need to parse the URL that was passed to us
// to extract the
// id and the content type.

$URI = $_SERVER['REQUEST_URI'];

if($URI == '/') {

    // If this is a request to root (/) redirect to page 1

    $request = array('pages',1);
    buildpage($request);

}else{

    // Parse the request, if it is valid get the content
    // from our DB

    $request = parse_request($URI);

    if(!is_null($request)){

        buildpage($request);

    }else{

        echo "Invalid request";

    }

}
```

Listing 4. *core.inc* replacement code

```
function buildpage($request) {

    // Content definitions
    require INCLUDES.'content.inc';

    // Routes our incoming request to the right content
    // type and pulls
    // the content from our DB.

    $content_type = $request[0];
    $id = $request[1];
    $template_file = TEMPLATES . $content_type . '_
    template.inc';

    // Build the SQL and get the result
```

```

$sql = "SELECT * FROM $content_type WHERE id='$id' LIMIT 1";
$result = mysql_select($sql);

// Check we have some content to display

if($result[0] == 0){

    echo 'No data';
    return;

}

// Check we have a template file

if(!file_exists($template_file)){

    echo 'No template';
    return;

}

// Don't write any output to browser yet as we want
to post process
// our content using a theme. If enabled use our
optimization
// callback to remove white space etc.

ob_start("optimize_callback");

// Output our page header

outfile(TEMPLATES . 'header.inc');

// Create our body

echo wraptag('title', $result['title']);
echo HEAD;
echo BODY;

// Generate a unique ID based on content type
// Map the requested content type from our real table name

$ct = array_search($content_type, $content_tables);

echo '<div id="' . $ct . '">';

// If we are in debug mode, show an alert

if(DEBUG){

    $theme['debug'] = div('&para;', '', 'debug');

}

// Dump the title & id out to our theme template
$theme['id'] = $result['id'];
$theme['title'] = $result['title'];

// As we don't know how many fields we will have in
our content
// type after our id, iterate through each in turn and wrap
// the field with a div

$offset = $result[1] - 1;
$pos = 0;

foreach($result as $key => $value){

    if($pos > $offset){

        $theme[$key] = div($result[$key], $key . '-' . $id, $key);

    }

    $pos ++;

}

// Add our standard copyright notice

$theme['licence'] = div(ahref(COPYRIGHT, LICENCE, 'Copyright and
licence details'), '', 'licence');

// Include our template file

require_once($template_file);

// Close our content type tag

echo '</div>';

// Output our HTML page footer
outfile(TEMPLATES . 'footer.inc');

// Flush it all out and display

ob_end_flush();

}

```

Listing 5. core.inc additional code

```

function parse_request($URI) {

    // Returns the type of content and the ID
    // of the content requested.
    // parse_request(/page/1)
    // $array['page'][1]
    // parse_request(/rubbish/123456)
    // NULL

    // Content definitions
    require_once INCLUDES.'content.inc';

    $ct = NULL;
    $id = NULL;
    $valid = 0;

    // Fetch the parameters from the URL

    $array = explode('/', $URI);

    // We don't need the first '/' - delete the first
    empty
    // array item

    $a = array_shift($array);

    // Check we have 2 parameters

    $paramcount = count($array);

    if($paramcount == 2){

        // First test passed - We have 2 parameters

        $valid ++;

        $ct = $array[0];
        $id = $array[1];
    }

    if(in_array($ct, $content_types)){

        // If content type matches our list second test
        passed

        $valid ++;

        // Map the requested content type to our real
        table name

        $array[0] = $content_tables[$ct];
    }

    if(is_numeric($id)){

        // If ID is a number, third test passed

        $valid ++;
    }

    if($valid == 3){

        // Valid parameters passed, return content type
        and page ID

        return $array;
    }else{

        // Test failed - return NULL

        return NULL;
    }
}

function optimize_callback($buffer){

    // Replace all spaces and cruft between tags

    if(OPTIMIZE){

        $b = preg_replace('~>\s+<~', '><', $buffer);
        $b = preg_replace('/\r\n|\r|\n/', '', $b);
        $b = preg_replace('!\s+!', ' ', $b);

        return $b;
    }
}

```

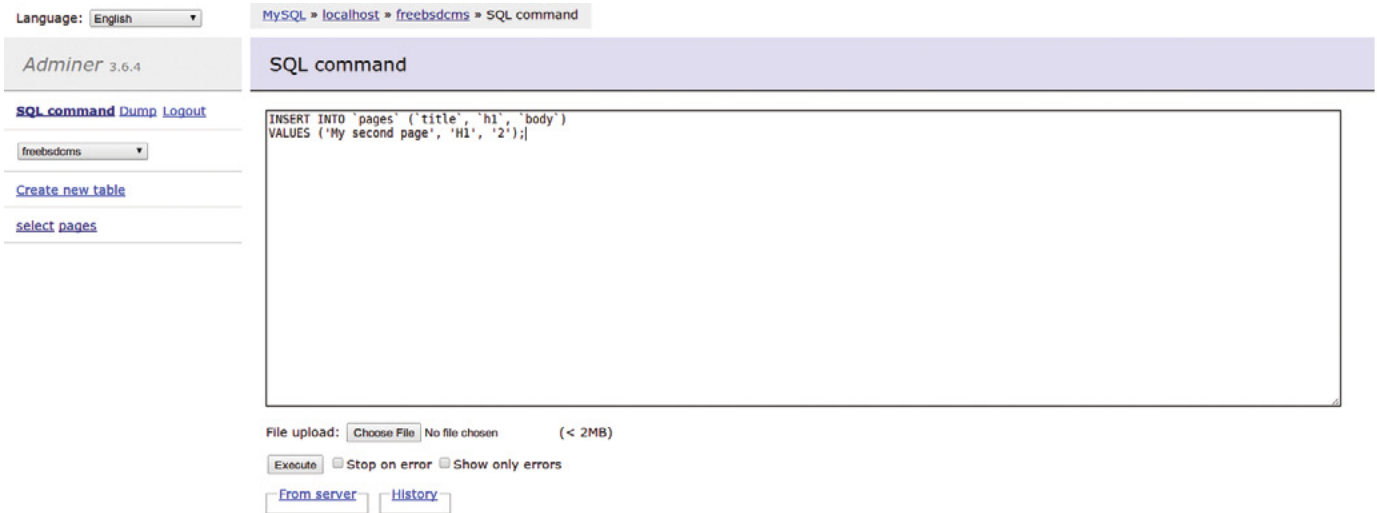


Figure 3. Using Adminer to execute SQL statement

Listing 6. *mysql.inc* replacement code

```
<?php
/*
 *
 * mysql.inc
 * Contains MySQL functions for our CMS
 *
 */
function mysql_select($sql) {
    $db = new mysqli(DBSERVER, DBUSER, DBPASSWORD, CMSDB);

    if($db->connect_errno > 0){
        die('Unable to connect to database [' .
            $db->connect_error . ']');
    }

    if(!$result = $db->query($sql)){
        if(DEBUG){
            die('There was an error running the query
            [' . $db->error . ']');
        }else{
            die('');
        }
    }

    // Pass our results to an array to be returned

    $r = array();

    $r[] = $result->num_rows;    // No of rows returned

    $r[] = $db->field_count;    // No of columns in table
    $r[] = $db->affected_rows;  // No of rows affected e.g.
                                // update / delete

    // Append the results to our result count

    if($result->num_rows != 0){
        $r = array_merge($r, $result->fetch_array(MYSQLI_
            ASSOC));
    }

    // Free the result

    $result->free();

    // Close the connection

    $db->close();

    return $r;
}
```


SPRING FUNDRAISING DRIVE

Your donations have helped make FreeBSD the best OS available! By investing in the services provided by The FreeBSD Foundation you have helped us fund projects to keep FreeBSD a high-performance, secure, and stable OS.

What will the Foundation accomplish with your donation in 2013?

- Software development projects for FreeBSD: \$600,000.
- Paid staff time supporting Release Engineering and Security teams.
- Grow staff: Five technical staff members by year-end.
- Provide support for BSD conferences around the globe, in Europe, Japan, Canada, and the USA.
- Hardware to maintain and improve FreeBSD project infrastructure: \$130,000.
- FreeBSD community growth through marketing and outreach to users and businesses.
- Legal services and counsel protecting the FreeBSD trademarks.



Support FreeBSD by donating

FreeBSD is internationally recognized as an innovative leader in providing a high-performance, secure, and stable operating system. Our mission is to continue and increase our support and funding to keep FreeBSD at the forefront of operating system technology. But, we can't do this without your help!

Last year with your generosity, we raised over \$770,000. This year we will invest \$1,000,000 to support and promote FreeBSD.

We have kicked off the new year with three newly funded projects, and are actively soliciting additional project proposals.

Please support the Foundation during our Spring Fundraising Drive, and help us raise \$100,000 from 1000 donors between April 15th and May 30th.

*We need your help.
We can't do this without you...*

Make your donation today. Go to:

www.freebsd.foundation.org/donate

The
FreeBSD
FOUNDATION

Then talk to your employer about matching your gift— or making their own donation.

Find out more, visit:

www.freebsd.foundation.org

types. We can then automatically use a custom template depending on the content type to post process our specific content.

First of all, we need to make some modifications to Apache so that it serves our index.php page as default. Edit the line in `/usr/local/etc/apache22/httpd.conf` to match the following:

```
DirectoryIndex index.php
```

Find the section marked `<Directory "/usr/local/www/apache22/data">` and add the following:

```
#
# Redirect on error via our CMS
#

ErrorDocument 401 /index.php
ErrorDocument 403 /index.php
```

Listing 7. *html.inc* replacement code

```
<?php
/*
 *
 * html.inc
 * Contains core html functions for our CMS
 *
 */

function wraptag($tag, $text) {

    // Wraps $text with compliant tags
    // wraptag('p',sometext)
    // <p>sometext</p>

    return '<' . $tag . '>' . $text . '</' . $tag . '>';
}

function div($divcontent, $class, $id = '') {

    // Generates a div tag $text with compliant tags
    // div('content','class')
    // <div class="class">content</div>
    // div('content','class','id')
    // <div id="id" class="class">content</div>
    // div('content','','id')
    // <div id="id">content</div>
    // div('content','','')
    // <div>content</div>

    if ($id != '') {

        $id = ' id="' . $id . '"';
    }

    if ($class != '') {

        $class = ' class="' . $class . '"';
    }

    return '<div' . $class . $id . '>' . $divcontent . '</div>';
}

function ahref($text, $url, $title = '') {

    // Generates an href tag $text with compliant tags
    // ahref('Click here',frebsd.org)
    // <a href="http://frebsd.org" title="Click here">Click here</a>
    // ahref('Click here',frebsd.org,'Link title')
    // <a href="http://frebsd.org" title="Link title">Click here</a>

    if ($title == '') {

        $title = $url;
    }

    $ahref = '<a href="' . $url . '" title="' . $title . '">' . $text . '</a>';

    return $ahref;
}

function render($field){

    // Renders via template

    echo $field;
}
```

```
ErrorDocument 404 /index.php
ErrorDocument 500 /index.php
```

This will force all traffic to be passed to our index.php for processing. As root, delete our unwanted files then re-start Apache:

```
$ rm /home/dev/data/index.xhtml
$ rm /home/dev/data/index.html
$ apachectl restart
```

When you visit <http://mysite> or <http://mysite/>, page 1 should be displayed. Now for the modifications that will facilitate content type routing and theme control. Create a file in the includes directory called *content.inc* with the content from Listing 1.

Create the following template file *pages_template.inc* in the templates directory shown in Listing 2.

Remove the following section entirely from index.php:

```
// Build page - use first record in database

$page['id'] = 1;
buildpage($page);
```

Replace with the one shown in Listing 3. Remove entirely the function call `buildpage($page)` from *core.inc*. Replace with the code shown in Listing 4. Add the function calls from Listing 5 to the end of *core.inc*.

Replace *html.inc* with Listing 7. Append the following to *cms.inc*:

```
// Optimize output by removing white space between tags etc.
define("OPTIMIZE", true);
```

Errata

In the previous article of this series the following syntax was incorrect:

```
#dev mysql -u root password 'cms-password' <
    createdb.sql
#dev mysql -u root password 'cms-password' < createpagetbl.sql
#dev mysql -u root password 'cms-password' < createpage.sql
```

It should have read:

```
#dev mysql -u root -p'cms-password' < createdb.sql
#dev mysql -u root -p'cms-password' < createpagetbl.sql
#dev mysql -u root -p'cms-password' < createpage.sql
```

Our apologies.

Useful Links

- SQL buddy – <http://sqlbuddy.com>
- Adminer – <http://www.adminer.org>

Testing and Adding New Content

That is a lot of code we have added, but we now have a major jump in functionality. We can create any number of content types now by creating a new table (e.g. *faq*, *news*, etc.) The only essential fields we must define are ID and TITLE. After these two fields you may define as many or as few as you require. You will need to create a matching template file with the fields you want to display or else the content will be unable to render. Once you have added new records to your content type (Adminer makes this quick and easy), the content can be accessed via your browser at: <http://mysite/mycontenttype/mypageid>. If you attempt to access invalid content, you will be presented with a rudimentary error message.

In the next article in the series, we will look at theming in detail and how we can lay out the site using a combination of templates and CSS.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

FreeBSD Programming Primer – Part 5

In the fifth part of our series on programming, we will look at how to apply a style using Cascading Style Sheets (CSS).

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

Developing a popular and effective web presence does not just rest on the back office technology per se, the website also requires “character” and often in corporate environments, strict style guidelines exist to ensure cohesive branding across media. Separating the design (in respect of “the look”) from the functional (what it “does”) remains a challenge for web developers. Quite often, the end result is a trade-off, especially when considering the number of web-enabled devices that are now available, the range of browser versions, font support and screen resolutions, etc. The bottom line is this – no matter how conscientious the web designer is, there are certain circumstances where the design of the website will not render as the designer expected.

The industry standard response to this is twofold. First, end users are encouraged to embrace newer browsers, thereby eliminating the more obvious compatibility issues, and secondly, designers look to format the site in such a way in that the visual output “degrades gracefully”

when approached by less compatible browsers. This scenario is further highlighted where Microsoft introduced compatibility mode in Internet Explorer 8 (available as an icon next to the refresh button) as it would not support the non-standard techniques used in previous versions of the browser.

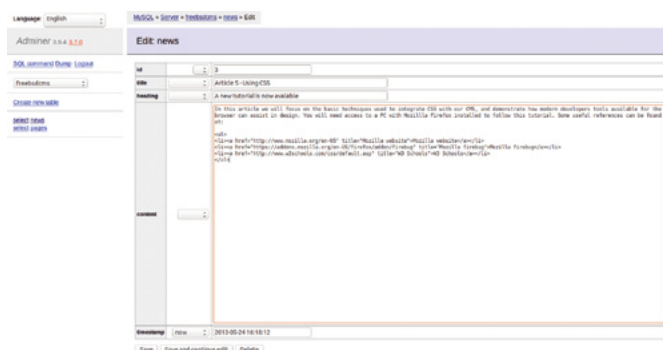


Figure 1. Adding new content via Adminer

Listing 1. Content added to content field via Adminer

In this article we will focus on the basic techniques used to integrate CSS with our CMS, and demonstrate how modern developers tools available for the browser can assist in design. You will need access to a PC with Mozilla Firefox installed to follow this tutorial. Some useful references can be found at:

```
<ul>
<li><a href="http://www.mozilla.org/en-US" title="Mozilla website">Mozilla website</a></li>
<li><a href="https://addons.mozilla.org/en-US/firefox/addon/firebug" title="Firebug">Firebug</a></li>
<li><a href="http://www.w3schools.com/css/default.asp" title="W3 Schools">W3 Schools</a></li>
</ul>
```


In this article, we will focus on the basic techniques used to integrate CSS with our CMS, and demonstrate how modern developers tools available for the browser can assist in design. You will need access to a PC with Mozilla Firefox installed to follow this tutorial.

Let's get started:

First of all, I have created a new news item (in this case, with an ID of 3) via the Adminer interface. I added the following content to the content field (the title and heading can be anything you want) – See (Listing 1) and (Figure 1).

If you point your browser at <http://youripaddress/news/3> you should see a web page similar to (Figure 2).

Firebug

When hand-crafting websites (i.e. when not using an util-



Figure 2. Our new news item



Figure 3. Firebug icon

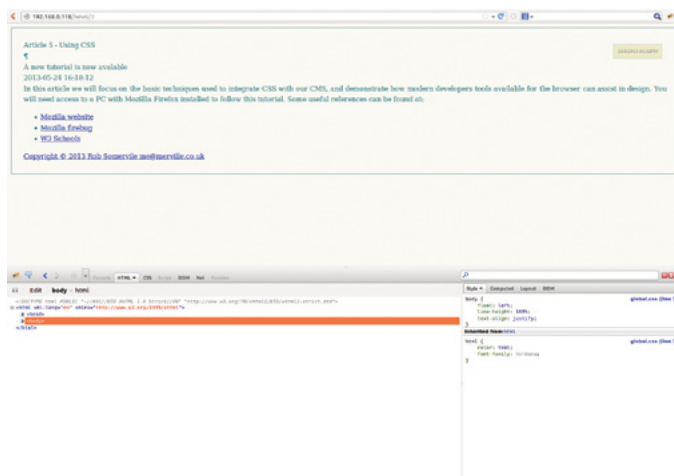


Figure 4. Using Firebug to view HTML code & modify CSS

ity such as Dreamweaver), one of the biggest headaches for the designer is writing CSS. The cycle goes like this – write CSS, refresh and preview in browser, correct mistakes, then repeat. With Firebug, we can view our CSS changes in real-time, adding selections and classes, etc. as required. The resulting amendments can be copied and pasted into our CSS file as required.

Either visit the Mozilla Firebug website available from the link or install Firebug via the Tools / Addons menu item and restart your browser. You should see the Firebug icon at the top right hand side (Figure 3). Clicking on the Firebug icon should bring up the Firebug interface and change the color of the icon (Figure 4).

If you click on the HTML, head or body tags in the left hand panel of Firebug, you will see the corresponding CSS as defined in our global.css file appear on the right. You can then disable or edit the values displayed as appropriate. Click on the HTML tag on the LHS and disable each CSS rule in turn by clicking next to it. To revert your changes, press F5. (Figure 5).

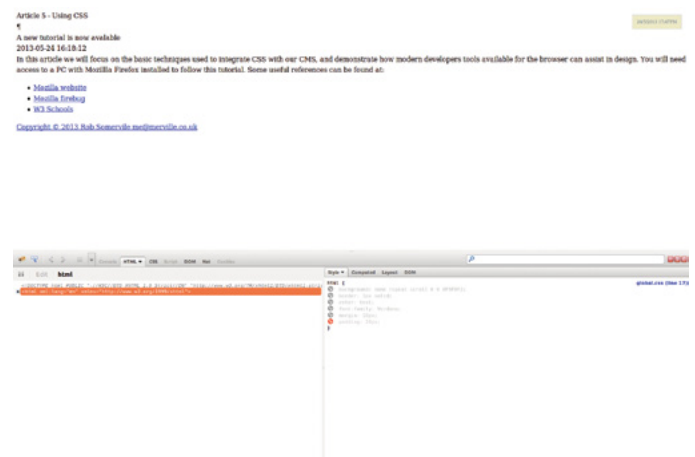


Figure 5. HTML attributes disabled in Firebug

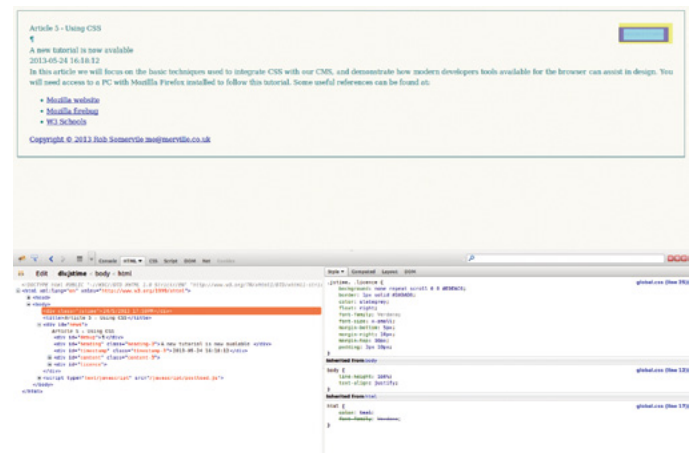


Figure 6. jstime div highlighted

Click on the HTML tab. Expand the body tag by clicking on the + sign, and click on `<div id="jstime">`. The div will be highlighted and the relevant CSS displayed. The corresponding CSS can be edited in situ as desired. See (Figure 6).

Firebug is not just an essential tool for modifying and testing CSS, you can also debug and step through Javascript, edit HTML on the fly, check how quickly the components of your web page download, etc.

Click on the Script tab and reload the page by pressing F5. The Javascript code from `preload.js` will be shown in the window. Set a breakpoint by clicking to the left of line 15 (`Var CurrentTime`) and reload the page again. You can then step through the javascript code by pressing F11. See (Figure 7).

Listing 2. *news_template.inc*

```
render($theme['heading']);
render($theme['title']);
render($theme['timestamp']);
render($theme['content']);
//render($theme['licence']);
//render($theme['debug']);
```

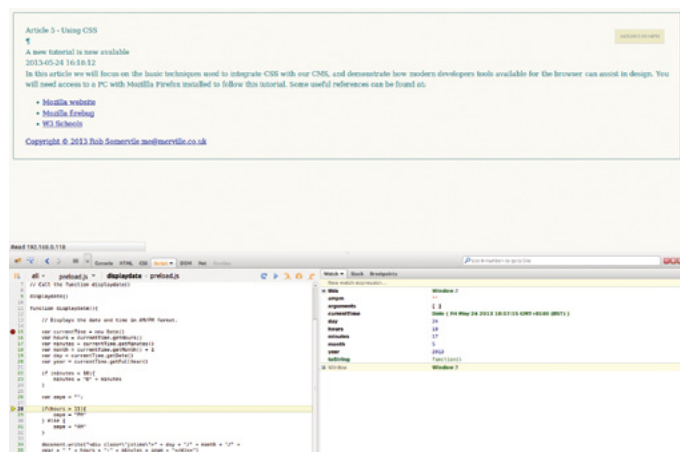


Figure 7. Debugging Javascript

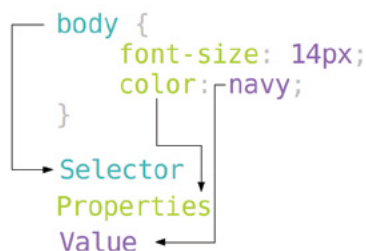


Figure 8. CSS selectors, properties and values

CSS selectors, properties and values

CSS syntax is very straightforward. Every HTML tag is referenced via a CSS selector (e.g. HTML → `html`, BODY → `body` etc.) and in turn, this selector has properties and values. Where matters get complicated is the cascading nature of CSS; for instance if the HTML is defined as having a font size of 12px (12 pixels), unless this is overridden somewhere, the body (and our footer areas) will have a font size of 12px. Good CSS is a balancing act between optimized selectors and overrides. Define your HTML or BODY too tightly and your theme will require lots of overrides. Likewise, if you have too loose a definition for your HTML or BODY tags, there will be a lot of unnecessary code duplication.

Cross browser compatibility also creates issues, developers often use a CSS reset to level the playing field to build on with their CSS.

As with all aspects of programming, there is never a definitive right answer. In some circumstances, speed will be more important than compatibility. In others, maximum compatibility will be more important and will require a lot more CSS. Complex designs will add to this payload.

Styling our site

We now have the following design requirements for our news page:

- The debug status symbol should be disabled for this page
- It should show the FreeBSD logo
- The content should be on a light background centered against a dark background
- The timestamp should be in a small font and should be prefaced with "Posted at"
- The heading should be in a large font and appear before the title
- List items should be discs
- All hyperlinks should be highlighted light red and change color on hover
- The content should leave a space on the RHS for some menu items to be added later
- The time display should be in the footer as well as the license conditions

First of all, download the FreeBSD logo from the FreeBSD site. This logo is a transparent PNG, which means whatever color background we display on the website will shine through the image. Download this into a new directory called `images` under `/dev/data`.

We need to make some modifications to our news template and javascript files. Edit `news_template.inc` to follow

Listing 3. header.inc

```
<link rel="stylesheet" type="text/css" href="//
  stylesheets/reset.css" />
```

Listing 4. footer_template.inc

```
<?php
  render($theme['licence']);
?>
```

Listing 5. cms.inc

```
// Definition of our footer template
define('TEMPLATE', 'footer_template.inc');
```

Listing 6. core.inc

```
// Include our template
require_once(TEMPLATES . FOOTERTEMPLATE);

// Output our HTML footer
```

the code in (Listing 2). This will change the order of the displayed items and disable the debug symbol.

Copy all the content from preload.js into postload.js, and delete everything apart from the opening comment from preload.js. Change the file title in the comments section of postload.js to postload.js. This will load the javascript clock at the end of the body section. See (Figure 9).

We now need to decide how global our CSS should be. Do we want all the links on the site to be light red? Should all list items be discs? Should the logo be the same on every page? For the sake of consistency, we will do this but can override this later via the theme files (by adding new divs, classes and id's) and adding further CSS.

Let us start off with a blank canvas. Delete the contents of global.css and download reset.css from meyerweb.com into the stylesheets directory. Add the following line immediately before we load global.css (Listing 3).

This will give us a page that looks like (Figure 10).

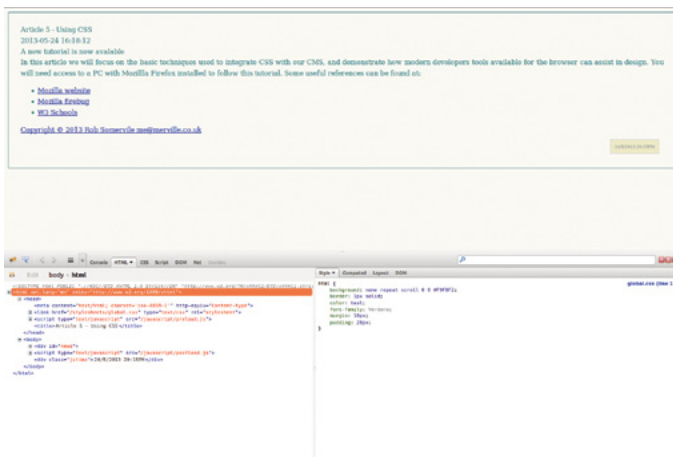


Figure 9. Re-ordered items and clock moved



Figure 11. Our re-branded news page

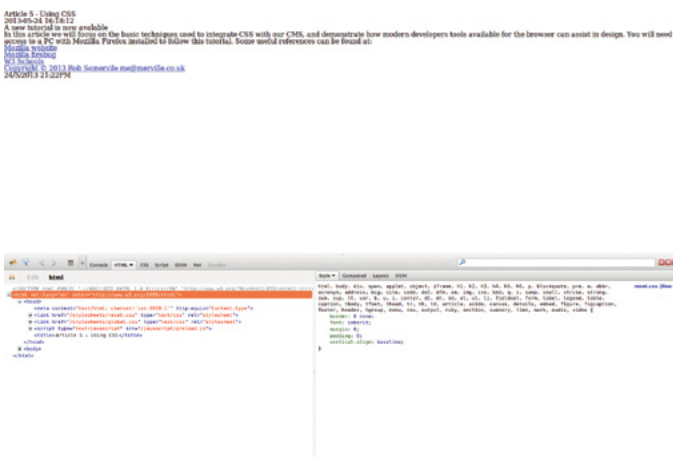


Figure 10. Site with reset.css applied

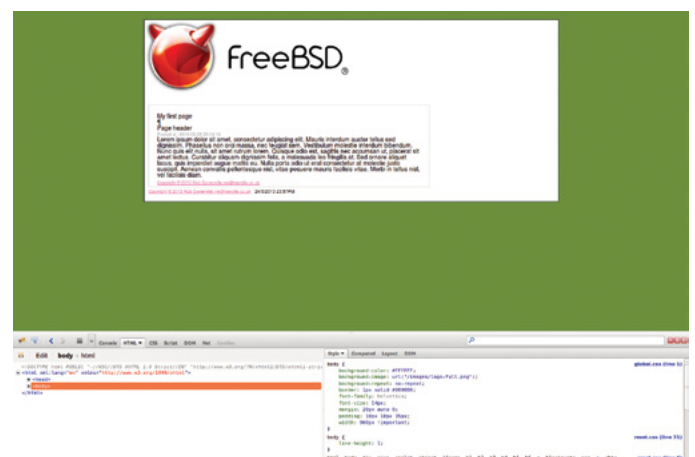


Figure 12. Front page – Note template needs amendment

Create the file `footer_template.inc` and add the following content (Listing 4). Add this to `cms.inc` (Listing 5).

Add the footer template to `core.inc` and add these lines just before `// Output our HTML footer` (Listing 6).

Create a new `global.css` with the following content (Listing 7). This should result in the pages as displayed in (Figure 11) and (Figure 12).

Table 1. *Global CSS*

Selector	Comments
body	No-repeat stops the image being repeated across the body area Margin and Width setting force the body to the centre !important overrides reset.css
a:hover	The: hover modifies the default a css to fire when the link is hovered over
#timestamp:before	Content added before the timestamp id
#heading	text-transform: capitalize forces the first letter of each heading word to uppercase
li	list-style:circle inside forces a disc to be used with indented list items

Listing 7. *global.css*

```
body {
    background-color: #FFFFFF;
    background-image: url("/images/logo-full.png");
    background-repeat: no-repeat;
    border: 1px solid #000000;
    font-family: helvetica;
    font-size: 14px;
    margin: 20px auto 0;
    padding: 10px 10px 35px;
    width: 960px !important;
}

html {
    background-color: olivedrab;
}

a {
    color: #FD5EA9;
}

a:hover {
    background-color: #FDA8D0;
    color: #FFFFFF;
}

#news, #page {
    border: 1px solid #DADADA;
    margin-top: 190px;
    padding: 20px;
    width: 65%;
}

#timestamp:before {
    content: "Posted at: ";
}

#timestamp {
    color: #A2A2A2;
    font-size: 10px;

    margin-top: 3px;
}

#heading {
    background-color: #B1348C;
    color: #FFFFFF;
    font-size: 25px;
    font-weight: bold;
    margin-bottom: 10px;
    margin-top: 10px;
    padding: 10px;
    text-transform: capitalize;
}

#content {
    color: #727272;
    font-size: 20px;
    line-height: 30px;
    text-align: justify;
}

#licence, .jstime {
    background: none repeat scroll 0 0 #FFFFFF;
    float: left;
    font-size: 10px;
    margin-top: 1px;
    padding-right: 10px;
    padding-top: 5px;
}

li {
    list-style: circle inside;
}
```




Useful links

- Mozilla firefox – <http://www.mozilla.org/en-US>
- Firebug – <https://addons.mozilla.org/en-US/firefox/addon/firebug>
- W3 Schools CSS tutorial – <http://www.w3schools.com/css/default.asp>
- FreeBSD logo – <http://www.freebsd.org/logo/logo-full.png>
- Eric Meyer's CSS reset – <http://meyerweb.com/eric/tools/css/reset/reset.css>

What have we accomplished here?

Apart from moving the javascript clock and licence outside of the main content area (which required the addition of a template and a small modification to core.inc and cms.inc), most of the heavy lifting has been performed by reset.css and global.css. Reset CSS sets the defaults for all major browsers etc, and this is reflected by the raw output on our non-news pages. We now have a choice, either embed our website standards in reset.css or depending on our database and template definitions, add some further CSS to global.css. With hindsight, choosing "body" for a field name in the pages table was not ideal, as it cannot be referenced in CSS without causing havoc. Renaming this to "content", changing the name in `pages_template.inc` and commenting out the `render($theme('licence'))` will fix the pages content to match the news item.

As for global.css, most of the selectors and values should be self-explanatory. The more subtle attributes are listed in (Box 1).

In the next article

We will continue with some more advanced CSS, and begin to build our menu system.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

**BSD development
and consultancy**

Zabbix Monitoring

**Bacula enterprise
backup**

BSD Thin Client

**Corporate BSD
Desktop**

**Solution
management
with Puppet**

and more ...

www.mtier.org
contact@mtier.org

FreeBSD Programming Primer – Part 6

In the sixth part of our series on programming, we will design a basic menu navigation system and style it with CSS.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

So far in this series, we have focused on adding and displaying standard HTML pages which have been pulled from our database. We are now going to shift directions and start to look at the user interface of the CMS itself. Traditionally, menu links were hard coded into pages, which not only made long-term maintenance time-consuming but also error-prone. By leveraging the power of a database back end, we can easily extract the title and section of pages we want to display and if desired, include or exclude that content from the menu. For flexibility, we will also include the facility to add disparate links to other sites, etc.

Many sites now use multi-level menus which are driven by a combination of SQL, Javascript / JQuery and CSS. Later on in the series, we will look at using JQuery to add this functionality, but for now we will concentrate on a block navigation menu that is displayed alongside the main content.

The SQL

To demonstrate, let's spin up a MySQL session and take a look at our content. At the shell prompt, login to MySQL and run some queries (Listing 1 – 2).

By using the UNION keyword, we can combine the output of both SELECT statements into one result. This would be fine if we had a small site with not much content, but as the site grows, the menu would become unmanageable in size. We could build the interface with a drop-

down and filter by section, but we would just be postponing the inevitable. An additional improvement would be to use a combination of a content type filter and a pager with the MySQL LIMIT keyword, restricting the display to a certain number of items. This would help in the final design

Listing 1. Logging in to MySQL

```
#dev mysql -u bsduser -pcmsdbpassword
```

Listing 2. Selecting our content

```
mysql> use freebsdcm;
mysql> (SELECT 'news' AS contenttype, id, title FROM
      news) UNION (SELECT 'pages' AS
      contenttype, id, title FROM pages);
```

```
+-----+-----+-----+
| contenttype | id | title |
+-----+-----+-----+
| news       | 1 | My first page |
| news       | 2 | My second page |
| news       | 3 | Article 5 - Using CSS |
| pages      | 1 | My first page |
| pages      | 2 | My second page |
+-----+-----+-----+
5 rows in set (0.00 sec)
```

and theming of the site, as we will know exactly how much browser real estate would be occupied by the menu itself even if the content expanded rapidly.

The remaining issues are how to add disparate links and whether we want to display the content in the menu at all. For example, we might have an error page that only is displayed when the content is not found. While it would be useful to store this in the database, displaying it in the menu would be rather pointless. The question is where to

store this data? We could have a separate menu table, with the ID of each page and a numeric flag (0, 1) to represent do not display in the navigation menu or include in the menu. We would then have to maintain 2 tables when content is added and removed. This could be easily accomplished using MySQL triggers. Alternatively, we could store the page status in the relevant content tables (e.g. news, pages) with a flag (0,1,2) to represent “do not publish”, “publish but do not show in menu”, and “publish and

Listing 3. Creating FAQ's table and adding status flag

```
mysql> CREATE TABLE faqs LIKE news;
mysql> ALTER TABLE faqs ADD status INT DEFAULT 0 AFTER
content;
```

Listing 4. Adding auto increment to the FAQ table

```
mysql> ALTER TABLE faqs CHANGE id id INT(11) AUTO_INCREMENT;
```

Listing 5. Adding data to the FAQ table

```
mysql> INSERT INTO faqs(id, title, heading, content,
status, timestamp) VALUES('',
'FAQ 1', 'First FAQ', 'Aenean volutpat, ligula vitae
laoreet dapibus',2,'');
```

Listing 6. Amending the remaining tables

```
mysql> ALTER TABLE pages ADD status INT DEFAULT 0 AFTER content;
mysql> ALTER TABLE news ADD status INT DEFAULT 0 AFTER content;
mysql> ALTER TABLE pages CHANGE id id INT(11) AUTO_INCREMENT;
mysql> ALTER TABLE news CHANGE id id INT(11) AUTO_INCREMENT;
```

Listing 7. Our 3 table content

```
mysql> (SELECT 'news' AS contenttype, id, status, title
FROM news) UNION (SELECT
'pages' AS contenttype, id, status, title FROM pages)
UNION (SELECT 'faqs' AS
contenttype, id, status, title FROM faqs);
```

```
+-----+-----+-----+-----+
| contenttype | id | status | title |
+-----+-----+-----+-----+
| news       | 1 | 0      | My first page |
| news       | 2 | 0      | My second page |
| news       | 3 | 0      | Article 5 - Using CSS |
| pages      | 1 | 0      | My first page |
| pages      | 2 | 0      | My second page |
| faqs       | 1 | 2      | FAQ 1 |
| faqs       | 2 | 0      | FAQ 2 |
| faqs       | 3 | 1      | FAQ 3 |
| faqs       | 4 | 2      | FAQ 4 |
```

```
| faqs       | 5 | 2      | FAQ 5 |
| faqs       | 6 | 2      | FAQ 6 |
| faqs       | 7 | 2      | FAQ 7 |
| faqs       | 8 | 2      | FAQ 8 |
| faqs       | 9 | 2      | FAQ 9 |
| faqs       | 10 | 2     | FAQ 10 |
```

```
+-----+-----+-----+-----+
15 rows in set (0.00 sec)
```

Listing 8. Updating the news and pages status

```
mysql> UPDATE news SET status = 1;
mysql> UPDATE pages SET status = 2;
mysql> (SELECT 'news' AS contenttype, id, status, title
FROM news) UNION (SELECT
'pages' AS contenttype, id, status, title FROM pages)
UNION (SELECT 'faqs' AS
contenttype, id, status, title FROM faqs);
```

```
+-----+-----+-----+-----+
| contenttype | id | status | title |
+-----+-----+-----+-----+
| news       | 1 | 1      | My first page |
| news       | 2 | 1      | My second page |
| news       | 3 | 1      | Article 5 - Using CSS |
| pages      | 1 | 2      | My first page |
| pages      | 2 | 2      | My second page |
| faqs       | 1 | 2      | FAQ 1 |
| faqs       | 2 | 0      | FAQ 2 |
| faqs       | 3 | 1      | FAQ 3 |
| faqs       | 4 | 2      | FAQ 4 |
| faqs       | 5 | 2      | FAQ 5 |
| faqs       | 6 | 2      | FAQ 6 |
| faqs       | 7 | 2      | FAQ 7 |
| faqs       | 8 | 2      | FAQ 8 |
| faqs       | 9 | 2      | FAQ 9 |
| faqs       | 10 | 2     | FAQ 10 |
```

```
+-----+-----+-----+-----+
15 rows in set (0.00 sec)
```



Figure 1. Bug in core.inc

show in menu". Both designs have their good and bad points from the implementation and data integrity viewpoint, but for the sake of simplicity, I will use the latter for our navigation menu.

In the meantime, we have an FAQ definition in our file content.inc but we do not have any table data for it. We will now manually create the table and add 10 random FAQ entries (Listing 3-5). This will result in a new FAQ table with our status field. However, the ID field is not set to auto increment, so we need to change this (Listing 4). Now insert the data (10 entries) – replacing the title, heading and status (0, 1 or 2) as appropriate. We need to repeat the structural amendments for our news and pages tables as well (Listing 6). Let's check what data we now have in the three tables (Listing 7). As we can see, the news and pages will not be published or displayed in the menu. Change this so the news items are not in the menu but published, but the pages are (Listing 8). Let us check in a browser if FAQ 1, 2 and 3 are displayed. Visit <http://yourserverip/faq/1> and



Figure 2. CSS requires fix for FAQ content type

you should get an error message "No template". To rectify this, create a `faqs_template.inc` file in `/usr/home/dev/data/templates` with the following content (Listing 9).

Bug alert! If you visit <http://yourserverip/faq/1> you will find the page is not rendering correctly (Figure 1). You will receive an error message: Notice: Undefined index: heading in `/usr/home/dev/data/templates/faqs_template.inc` on line 23. If you want to try and diagnose the problem, have a look at `core.inc` and skip the next code listing. The problem lies in the following code snippet. To fix it, change as follows (Listing 10-11).

If you visit <http://yourserverip/faq/1>, you will find the page is still not rendering correctly (Figure 2). The reason for this is that the the global CSS doesn't know about our FAQ content type yet, so we need to modify `global.css` as follows (Listing 12). You may have to refresh or clear your browser cache to pick this up. This should result in the

Listing 9. FAQ template

```
<?php
/*
 *
 * faqs_template.inc
 * Template for our faq content type
 *
 * For content type foo the corresponding template would be:
 * foo_template.inc
 *
 * To display a field:
 * render($theme['name_of_field_as_defined_in_db']);
 *
 * To hide a field omit it from here
 * To change the rendering order, just re-order the fields
 *
 * NOTE: Any content generated by javascript will not
 * be managed here
 * A closing >? tag is mandatory
 */
render($theme['heading']);
render($theme['content']);
```

```
>?
```

Listing 10. Bad code!

```
if($pos > $offset){

    $theme[$key] = div($result[$key], $key.'-'. $id, $key);

}
```

Listing 11. Good code

```
if($pos >= $offset){

    $theme[$key] = div($result[$key], $key.'-'. $id, $key);

}
```

Listing 12. CSS to include FAQ content type

```
#news, #page, #faq {
```

Listing 13. Prevent non-published content showing

```
$sql = "SELECT * FROM $content_type WHERE id='$id' AND
status > 0 LIMIT
1";
```



Figure 3. FAQ working

correctly rendered content in (Figure 3). However, if we visit `http://yourserverip/faq/2`, we will see an FAQ page even though the status is 0. Modify `core.inc` as follows to fix this (Listing 13). This should now give a “No data” message. If you are still experiencing problems, ensure that the `content.inc` file is as follows (Listing 14).

Building our menu

How can we remember the filter value selected for the content type? As HTTP is stateless, we could pass the parameter to each page. This would get complex very quickly with multiple menus. A better solution would be to write a cookie to the visitors browser when the content type is filtered. To do this we will use Javascript, and specifically a suite of JQuery

Useful links

- JQuery library: <http://code.jquery.com/jquery-1.10.2.min.js>
- JQuery cookie: <https://github.com/carhartl/jquery-cookie/blob/master/jquery.cookie.js>

libraries. Download `jquery-1.10.2.min.js` and `jquery.cookie.js` from the JQuery website. Place these files in the Javascript folder, then modify our source code as follows (Listing 15-18). When you visit `http://youripaddress/faq/1`, you should see a page similar to Figure 4. Clicking on the FAQ, News or Page button will raise a Javascript dialogue box.

In the next part

We will tie the onclick event to writing a local cookie, and extracting the links for the MySQL table. We will also look at using the JQuery library to build a multi-part menu.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

Listing 14. `content.inc`

```
<?php
/*
 *
 * content.inc
 * Defines content types for our CMS
 *
 */

// Define the content type. This must match any tables
// defined in our
// CMS

$content_types[] = 'page';
$content_types[] = 'faq';
$content_types[] = 'news';

// Map each content type to a table. Each content type
// must be matched
// to a corresponding table
```

```
$content_tables['page'] = 'pages';
$content_tables['faq'] = 'faqs';
$content_tables['news'] = 'news';
```

Listing 15. `header.inc` include JQuery support

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-type" content="text/html;
charset='iso-8859-1'" />
<link rel="stylesheet" type="text/css"
href="/stylesheets/reset.css" />
<link rel="stylesheet" type="text/css"
href="/stylesheets/global.css" />
<script src="/javascript/jquery-1.10.2.min.js"
type="text/javascript"></script>
<script src="/javascript/jquery.cookie.js"
type="text/javascript"></script>
<script src="/javascript/preload.js" type="text/javascript">
</script>
```


Listing 16. core.inc

```
function optimize_callback($buffer) {

    // replace all spaces between tags

    if (OPTIMIZE) {

        $b = preg_replace('~>\s+<~', '><', $buffer);
        $b = preg_replace('~\r\n|\n\r|\n/', '', $b);
        $b = preg_replace('~!\s+!~', ' ', $b);

        return $b;

    } else {

        // BUGFIX - Edition 6

        return $buffer;

    }
}
```

Listing 17. index.php – add include menu.inc

```
// Menu functions
require_once INCLUDES.'menu.inc';
```

Listing 18. menu.inc

```
<?php

function menu($type) {

    require INCLUDES . 'content.inc';

    if ($type == 'navigation') {

        // Build select statement for each content type
        in turn
        // Omit the UNION keyword on the last item

        $offset = 1;
        $categories = count($content_tables);
        $sql = '';
        $option = '';

        foreach ($content_tables as $contenttype) {

            // Build the option for the content type

            $option .= '<button onclick="window.alert(\''.
                $contenttype.'\')">'. $contenttype.' &nbsp;';
```

```
$offset ++;

        }

        $menu = '';

        $menu .= '<div class="menu-' . $type . '">';
        $menu .= '<h2>' . $type . '<h2>';
        $menu .= '<p>&nbsp;</p>';
        $menu .= $option;
        $menu .= '</div>';

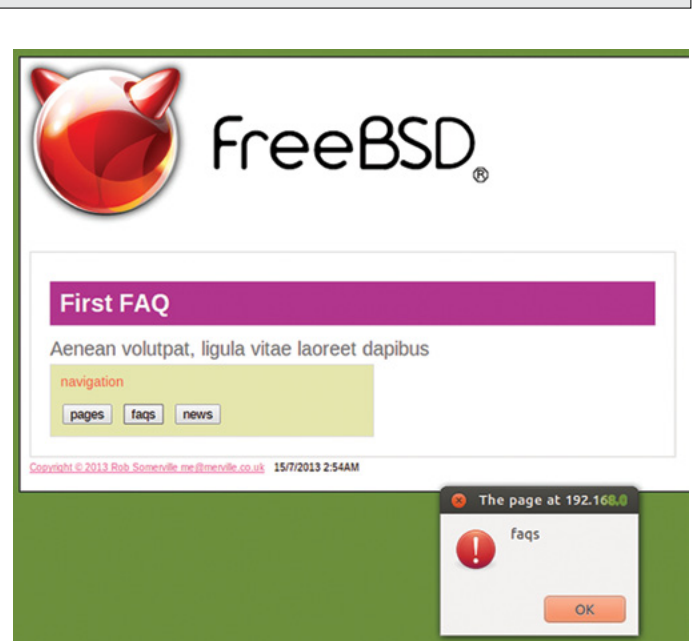
        return $menu;

    }
}
```

Listing 19. Menu CSS add to global.css

```
.menu-navigation {
    border: 1px solid #DADADA;
    padding: 10px;
    width: 50%;
    background-color: #E5E6AD;
}

h2 {
    color: tomato;
    font-weight: 600;
}
```

**Figure 4. FAQ with Javascript onclick buttons**

Is your
MISSION-CRITICAL
security strong enough
to stop a
SKILLED ATTACKER?

Don't guess
Don't believe
Don't hope

KNOW!



An ACROS Penetration Test is **conducted exactly like a real attack by a skilled, motivated adversary** – only without the damage. We will find the weakest links in your security and use all our knowledge, skills and capabilities to try to achieve exactly what your security measures and policies are there to prevent.
If it sounds difficult, we're interested.

Experience **the ultimate test of your security.**
(After all, the only alternative is to wait for an actual attack.)

FreeBSD Programming Primer – Part 7

In the seventh part of our series on programming, we will continue with the menu navigation system and using Javascript.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

So far, we have built navigation section buttons that represent the three content types that we have defined in content.inc: pages, news and FAQ's. When

the button is pressed, a Javascript popup alerts the user as to what button was clicked via the onclick event (Figure 1). We now need to add additional functionality – when the page is loaded, by default the page's links should be displayed, the menu option (or filter) needs to be displayed to the user, and when the button is clicked, the menu content needs to be rebuilt (Figure 2). Later we will build a more sophisticated menu using the JQuery library.



Figure 1. FAQ with Javascript onclick buttons

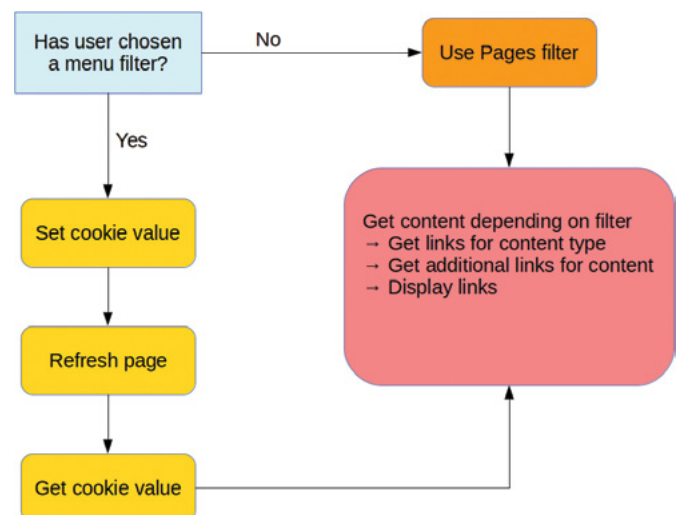


Figure 2. Logic for the navigation menu

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd":
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
3 <head>
4 <meta http-equiv="Content-type" content="text/html; charset='iso-8859-1'" />
5 <link rel="stylesheet" type="text/css" href="/stylesheets/reset.css" />
6 <link rel="stylesheet" type="text/css" href="/stylesheets/global.css" />
7 <script src="/javascript/jquery-1.10.2.min.js" type="text/javascript"></script>
8 <script src="/javascript/jquery.cookie.js" type="text/javascript"></script>
9 <script src="/javascript/preload.js" type="text/javascript">
10 </script>
11 <title>My first page</title></head><body><div id="page">My first page<div id="debug">&para;</div><div id="h1
12 interdum auctor tellus sed dignissim. Phasellus non orci massa, nec feugiat sem. Vestibulum molestie interdum
13 bibendum. Nunc quis elit nulla, sit amet rutrum lorem. Quisque odio est, sagittis nec accumsan ut, placerat :
14 amet lectus. Curabitur aliquam dignissim felis, a malesuada leo fringilla at. Sed ornare aliquet lacus, quis
15 imperdiet augue mattis eu. Nulla porta odio ut erat consectetur at molestie justo suscipit. Aenean convallis
16 pellentesque nisl, vitae posuere mauris facilisis vitae. Morbi in tellus nisl, vel facilisis diam.</div></di
17

```

Figure 3. Page source showing Javascript JQuery libraries loaded

Listing 1. *postload.js*

```

// Set navigation menu cookie                                $menuvalue = 'pages';

function setnavitem(item){                                    }

$.cookie("navmenuitem", item);                                foreach ($content_tables as $contenttype) {

}                                                            // Build the option for the content type

```

Listing 2. *menu.inc*

```

<?php                                                        $option .= '<button onclick="setnavitem(\''.
                                                            $contenttype.\')';
                                                            document.location.reload(true);">'. $offset.'.
                                                            \'.ucfirst($contenttype).\'/>button> &nbsp;';
                                                            $offset ++;

function menu($type) {                                        }

    require INCLUDES . 'content.inc';

    if ($type == 'navigation') {                                $menu = '';

        // Build select statement for each content type
        in turn                                                $menu .= '<div class ="menu-' . $type . '">';
        // Omit the UNION keyword on the last item              $menu .= '<h2>' . ucfirst($type) . ' (' .
                                                            $menuvalue.\')</h2>';
                                                            $menu .= '<p>&nbsp;</p>';
                                                            $menu .= $option;
                                                            $menu .= '</div>';

        $offset = 1;                                            return $menu;
        $categories = count($content_tables);
        $sql = '';
        $option = '';

        // Get the value of the cookie if set                    }

        if(isset($_COOKIE['navmenuitem'])) {
            $menuvalue = $_COOKIE['navmenuitem'];
        } else {

```

Step 1 – Handling the user interaction

Ensure you have downloaded the JQuery libraries as detailed in the previous article. If you view the page source



Figure 4. FAQ page with Javascript and cookie control



Figure 5. Cookie set in Firebug

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
2 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
3 <head>
4 <meta http-equiv="Content-type" content="text/html; charset='iso-8859-1'" />
5 <link rel="stylesheet" type="text/css" href="/stylesheets/reset.css" />
6 <link rel="stylesheet" type="text/css" href="/stylesheets/global.css" />
7 <script src="/javascript/jquery-1.10.2.min.js" type="text/javascript"></script>
8 <script src="/javascript/jquery.cookie.js" type="text/javascript"></script>
9 <script src="/javascript/preload.js" type="text/javascript">
10 </script>
11 <title>FAQ 1</title></head><body><div id="faq"><div id="heading" class="heading-1">First FAQ</div><div
id="content" class="content-1">Aenean volutpat, ligula vitae laoreet dapibus</div><div class="menu-navigation">
<h2>Navigation (pages)</h2><p>&nbsp;</p><p><button onclick="setnavitem('pages'); document.location.reload(true);">1.
Pages</button> &nbsp;<button onclick="setnavitem('faqs'); document.location.reload(true);">2. Faqs</button> &nbsp;<button
onclick="setnavitem('news'); document.location.reload(true);">3. News</button> &nbsp;</div></div><div
id="licence"><a href="licence.txt" title="Copyright and licence details">Copyright &copy; 2013 Rob Somerville
me@merville.co.uk</a></div><script src="/javascript/postload.js" type="text/javascript"></script></body></html>

```

Figure 6. Page source showing button options

for any page, it should be similar to (Figure 3). Modify postload.js and menu.inc as follows (Listing 1 – 2).

If you now navigate to <http://yoursiteip/faq/1>, you should now see a page similar to (Figure 4). If you click on the buttons, instead of a Javascript popup you should see the navigation menu title changing to reflect the new selection. Using Firebug and the Cookie console, you will see the content of the cookie changing when a new menu item is selected. Deleting the cookie and refreshing the

Listing 3. add to preload.js

```

// Init routines

function preinit(){

    document.body.style.display = 'none';

}

function postinit(){

    $(document.body).fadeIn(500);

}

```

Listing 4. Add to core.inc

```

Add just after echo BODY;

echo "<script>preinit();</script>";

```

Listing 5. Add to core.inc

```

Add just before ob_end_flush()

echo "<script>postinit();</script>";

```


Listing 6. *Add to mysql.inc*

Returns an array of rows or **NULL** on no result

```
function mysql_fetchrows($sql) {

    // Returns an array of rows or NULL if no result

    $db = new mysqli(DBSERVER, DBUSER, DBPASSWORD,
CMSDB);

    if ($db->connect_errno > 0) {
        die('Unable to connect to database [' .
$db->connect_error . ']');
    }

    if (!$result = $db->query($sql)) {
        if (DEBUG) {
            die('There was an error running the query [' .
. $db->error . ']');
        } else {
            die('');
        }
    }

    while ($row = $result->fetch_row()) {
        $r[] = $row;
    }

    // Free the result

    $result->free();

    // Close the connection

    $db->close();

    if (isset($r)) {
        return $r;
    } else {
        return NULL;
    }
}
```

Listing 7. *Add to core.inc*

```
function arraytolinks($mysqlfetchrows){

    require INCLUDES . 'content.inc';

    // Convert a MySQL result set into a set of links
```

```
    // Requires ID (page id), title and contenttype

    $links = '<div class="menulinks">';
    $links .= '<ul>';

    if($mysqlfetchrows){

        foreach ($mysqlfetchrows as $key => $value) {

            // Convert the content type to the relevant
table name.
            // See content.inc

            $path = array_search($value[2], $content_
tables);

            $links .= '<li><a href="/" . $path . "/" . $value[0].'"
title="' . $value[1].'">'.
                $value[1]. '</a></li>';

        }

        $links .= '</ul>';
        $links .= '</div>';

    }else{

        $links .= "<li>Sorry - no content available</
li></ul></div>";

    }

    return $links;
}
```

Listing 8. Full listing of menu.inc

```
<?php

function menu($type) {

    require INCLUDES . 'content.inc';

    if ($type == 'navigation') {

        $offset = 1;
        $categories = count($content_tables);
        $sql = '';
        $option = '';

        // Get the value of the cookie if set

        if(isset($_COOKIE['navmenuitem'])){
            $menuvalue = $_COOKIE['navmenuitem'];
        }else{
            $menuvalue = 'pages';
        }

        foreach ($content_tables as $contenttype) {

            // Build the option for the content type

            $option .= '<button onclick="setnavitem(\''.
$contenttype.'\')';
            document.location.reload(true);">'. $offset.'.'.
            '.ucfirst($contenttype).'

```

```
        $menu .= '<div class ="menu-' . $type . '">';
        $menu .= '<h2>' . ucfirst($type) . ' (' .
$menuvalue.') - '.$categories.'
        categories</h2>';
        $menu .= '<p>&nbsp;</p>';
        $menu .= $option;
        $menu .= $links;
        $menu .= '</div>';

        return $menu;
    }
}
```

Listing 9. Changes to faq_temptate.inc

```
render($theme['heading']);
render(menu('navigation'));
render($theme['content']);
```

Listing 10. Modify global.css

```
.menu-navigation {
    background-color: #E5E6AD;
    border: 1px solid #DADADA;
    padding: 10px;
    float: right;
    margin-left: 10px;
    margin-bottom: 10px;
}

#news, #page, #faq {
    border: 1px solid #DADADA;
    margin-top: 190px;
    padding: 20px;
    min-height: 640px;
    overflow: auto;
}
```

Listing 11. Add global menu support to News, FAQ and pages templates

Add at the beginning of each file (e.g. just before
render(\$theme['heading']));

```
render(menu('global'));
```

Listing 12. Add to preload.js

```
function globalmenu(){

    $(function() {$( "#menu" ).menu();});

}
```

Listing 13. header.inc

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">
<head>
<meta http-equiv="Content-type" content="text/html; charset='iso-8859-1'" />
<link rel="stylesheet" type="text/css" href="/stylesheets/reset.css" />
<link rel="stylesheet" type="text/css" href="/stylesheets/global.css" />
<link rel="stylesheet" type="text/css" href="/stylesheets/jquery-ui.css" />
<script src="/javascript/jquery-1.10.2.min.js" type="text/javascript"></script>
<script src="/javascript/jquery.cookie.js" type="text/javascript"></script>
<script src="/javascript/jquery-ui.min.js" type="text/javascript"></script>
<script src="/javascript/preload.js" type="text/javascript">
</script>
```

page should load the default menu type of Pages (Figure 5). The titles have also been cleaned up using the PHP `ucfirst()` function call to upcase the first character of the selection, and we have added a sequential option number to each menu item.

One disadvantage of this method is the following piece of code as shown in (Figure 6). Each button has two pieces of Javascript attached, `setnavitem()` and `document.location.reload()`. The former sets the cookie via our function call in `postload.js` (and subsequently via the `jquery.cookie.js` script) and then refreshes the page. This causes the page to flicker every so often when the con-

tent is reloaded. A better way of implementing this would be to use Ajax, but for the time being, we will demonstrate a useful JQuery call – Fade in.

Add the following code to `preload.js` (Listing 3) and `core.inc` (Listing 4 and Listing 5).

This will halt the display of the page, allow the menu to be built etc. and the page will then fade in. The time can



Figure 7. FAQ page menu



Figure 8. FAQ faqs menu



Figure 9. FAQ news menu



Figure 10. JQuery multi-level menu

be adjusted by incrementing or decrementing the `fadeIn()` parameter. While this is not an ideal solution, it does demonstrate the ease of integrating JQuery with a web page.

Step 2 – Displaying the links

Now we need to plug in the SQL result to our menu module. Add the following code (Listing 6-8).

We now need to make a few minor modifications at the theme and CSS levels, so change `faq_template.inc` to display the menu before the content (Listing 9).

Listing 14. Additions to `menu.inc`

```
Add elseif at the end of the navigation block

$menu .= '<div class ="menu-' . $type . '">';
$menu .= '<h2>' . ucfirst($type) . ' (' .
$menuvalue.') - '.$categories.'
categories</h2>';
$menu .= '<p>&nbsp;</p>';
$menu .= $option;
$menu .= $links;
$menu .= '</div>';

return $menu;

}elseif ($type == "global") {

    ?>

    <ul id="menu">
        <li><a href="/">Home</a>
        <ul>
            <li><a href="/page/1">Pages</a></li>
            <li><a href="/news/1">News</a></li>
            <li><a href="/faq/1">FAQ's</a></li>
        </ul>
        </li>
    </ul>

    <?php

}
```

Listing 15. Add to `global.css`

```
.ui-menu {
    width: 150px;
}
```

Useful links

- JQuery UI source – <http://jqueryui.com/resources/download/jquery-ui-1.10.3.zip>
- JQuery menu reference – <http://jqueryui.com/menu>

This will float the navigation menu on the FAQ page to the right and increase the height of our news, page, and FAQ content to accommodate the new menu.

See (Figure 7-9) for the final result. I added an extra “Ipsum Lorem” to pad the content out in FAQ 3. Note how the menu responds to user input decoupled from the content that the user is currently visiting.

Step 3 – Global website menu

Jquery provides an extensive library for the user interface. Rather than building the Javascript and CSS from scratch, we can install the CSS and JS libraries quickly into our CMS.

Download JQuery-ui-1.10.3.zip and extract JQuery-ui.css into the stylesheets directory and JQuery-ui.min.js into the javascript directory. Use MC, or extract the file into a temporary area using unzip.

Add the global menu to all of our content templates (`news_templates.inc`, `pages_template.inc` and `faqs_template.inc`) and add the Javascript function to `preload.js`. Add the Javascript and CSS files to the `header.inc` file and add a new menu option to `menu.inc` and finally tweak our CSS file to reduce the width of the menu (Listing 11-15).

Finally, visit the homepage of your site with your browser, refresh the page and voila, one multi-level menu (Figure 10).

In the next part

We will continue refining the menu system and start building the user interface.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.



IT'S NOT ABOUT DATA. IT'S ABOUT MEANING.



If you think mobile forensics is just about extracting data – think again. Its not only what you get, but what you do with it that really makes the difference.

XRY has an intuitive GUI that's easier to use, has better display capabilities and superior analysis functionality.

FreeBSD Programming Primer – Part 8

In the eighth part of our series on programming, we will refine our JQuery menu and start building a user friendly interface to add content.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

In the previous article, we implemented a menu using the JQuery library. Looking at menu.inc, we see the menu is “hard coded” with a top level menu Home, and sub menu’s Pages, News and FAQ’s. To make our CMS user friendly, ideally we would store the menu values in a database table that we could access and amend from a web form (Listing 1 and Figure 1).

Rather than building a custom page for each table, it would be good practice to design a set of global functions (e.g. sign on, retrieve fields, save fields etc.) and design a

template that we could change on a per table / form basis. We could then quickly build forms to modify each type of content. We also need to tweak the CSS for our dropdown menu. At the moment with the default CSS, the menu is floating to the left hand side. We will modify this to accommodate a wider menu with more options.

Step 1

For the initial testing, we will hand code a menu in menu.inc and add a few placeholders. Once we are happy with the CSS, we will then add this to a database table and add our forms. In the next article, we will write the code to extract the menu values and fire them into JQuery.



Figure 1. Original JQuery menu

Listing 1. menu.inc

```
<ul id="menu">
<li>
<a href="/">Home</a>
<ul>
<li><a href="/page/1">Pages</a></li>
<li><a href="/news/1">News</a></li>
<li><a href="/faq/1">FAQ's</a></li>
</ul>
</li>
</ul>
```



Figure 2. JQuery menu horizontal



Figure 3. JQuery menu with drop down menu

Listing 2. Replacement JQuery menu

```
<div id="jquerymenu">
  <ul id="top-menu-home">
    <li><a href="/">Home</a></li>
  </ul>

  <ul id="top-menu-pages">
    <li><a href="/">Pages</a>
      <ul>
        <li><a href="/page/1">Page 1</a></li>
        <li><a href="/page/2">Page 2</a></li>
        <li><a href="/page/3">Page 3</a></li>
      </ul>
    </li>
  </ul>

  <ul id="top-menu-news">
    <li><a href="/">News</a>
      <ul>
        <li><a href="/news/1">News 1</a></li>
        <li><a href="/news/2">News 2</a></li>
        <li><a href="/news/3">News 3</a></li>
      </ul>
    </li>
  </ul>

  <ul id="top-menu-faq">
    <li><a href="/">FAQ's</a>
      <ul>
        <li><a href="/faq/1">FAQ 1</a></li>
        <li><a href="/faq/2">FAQ 2</a></li>
        <li><a href="/faq/3">FAQ 3</a></li>
      </ul>
    </li>
  </ul>
```

```
<ul id="top-menu-user">
  <li><a href="/login.php">Login</a></li>
</ul>
</div>
```

Listing 3. preload.js

```
function globalmenu() {

  $(function() {$( "#top-menu-home" ).menu();});
  $(function() {$( "#top-menu-pages" ).menu();});
  $(function() {$( "#top-menu-news" ).menu();});
  $(function() {$( "#top-menu-faq" ).menu();});
  $(function() {$( "#top-menu-user"
    ).menu();});

}
```

Listing 4. global.css

```
#jquerymenu {
  border: 1px solid #DADADA;
  margin-bottom: 10px;
  height: 48px;
  padding: 5px;
  background-color: #e8e7cf;
}

.ui-menu{
  float: left;
}
```

Listing 5. create the menus table

```
CREATE TABLE `menus` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `group` varchar(12) NOT NULL,
  `menutitle` varchar(12) NOT NULL,
  `titleurl` varchar(12) DEFAULT NULL,
  `submenutitle` varchar(50) DEFAULT NULL,
  `submenutitleurl` varchar(50) DEFAULT NULL,
  `order` int(2) NOT NULL DEFAULT '0',
  `enabled` int(1) NOT NULL DEFAULT '1',
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

Listing 6. populate the menus table

```
INSERT INTO `menus` (`id`,`group`,`menutitle`,`titleurl`,
  `submenutitle`,
  `submenutitleurl`,`order`,`enabled`,`timestamp`) VALUES
(1,'jquerymenu','Home','/',NULL,NULL,1,1,'2013-09-02
  17:50:05'),
(2,'jquerymenu','Pages',NULL,NULL,NULL,2,1,'2013-09-02
  17:54:58'),
(3,'jquerymenu','Pages',NULL,'Page 1','/
  page/1',1,1,'2013-09-02 17:56:33'),
(4,'jquerymenu','Pages',NULL,'Page 2','/
  page/2',2,1,'2013-09-02 17:57:28'),
(5,'jquerymenu','Pages',NULL,'Page 3','/
  page/3',3,0,'2013-09-02 17:58:08');
```

Listing 7. amendcontentpage.php

```
<?php
```

```
require_once 'includes/cms.inc';
require INCLUDES . 'content.inc';
require INCLUDES . 'core.inc';
require INCLUDES . 'html.inc';
require INCLUDES . 'mysql.inc';

// SQL statements

$sql[0] = "SELECT COUNT(DISTINCT TABLE_NAME) FROM
  INFORMATION_SCHEMA.COLUMNS
  WHERE table_schema = 'freebsdcsms'
  AND TABLE_NAME = '---P0---'";
$sql[1] = "SELECT
  TABLE_NAME,COLUMN_NAME,COLUMN_DEFAULT,IS_
  NULLABLE,DATA_TYPE,
  CHARACTER_MAXIMUM_LENGTH
```

```
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_schema = 'freebsdcsms'
AND TABLE_NAME = '---P0---'
ORDER BY table_name, ordinal_position";
```

```
// The tables we will allow the user to edit via this
form
```

```
$tables[] = "faqs";
$tables[] = "menus";
$tables[] = "news";
$tables[] = "pages";
```

```
// Fields that are automatically assigned via a default
value in MySQL table
// definition
```

```
$skiplist[] = "id";
$skiplist[] = "timestamp";
```

```
////////////////////////////////////
////////////////////////////////////
////////////////////////////////////
```

```
// Build the page up to the body tag
```

```
outfile(TEMPLATES . 'header.inc');
echo wraptag('title', 'Content Input');
echo HEAD;
echo BODY;
```

```
// Page control logic
```

```
if(isset($_POST["table"])){
```

```
// User has not selected a table or we are testing
their result
```

```
$t = $_POST["table"];
```

```
if (!in_array($t, $tables)) {
```

```
// If the table is not on allowed list, bail to
the first page
```

```
build_page_1($tables);
```

```
}else{
```

```

// Check selected table is valid

$s = $sql[0];

// Replace the marker in the SQL statement with
the chosen value
$s = str_replace ( '---P0---' , $t , $s );

$result = mysql_select($s);
$valid_table_count = $result['COUNT(DISTINCT
TABLE_NAME)'];

if($valid_table_count == 1){

    // Valid table selected - present form to
    edit data
    build_page_2($t,$sql,$skiplist);

}else{

    // Send user to first page
    build_page_1($tables);

}

}

}elseif(isset($_POST["update"])){

    // Save the input. As we have not validated this,
    just display for now

    build_page_3($_POST);

}else{

    // Invalid value - return to start

    build_page_1($tables);

}

////////////////////////////////////////
////////////////////////////////////////
////////////////////////////////////////

function build_page_1($tables){

    // HTML form definition

    echo '<div id="content">';
    echo '<div id="php">';
    echo '<div id="h1">1: Select content</div>';
    echo '<form action="amendcontent.php" method="post">';
    echo '<select name="table">';

    foreach ($tables as $t){

        // $tables is an array - split each value out

        echo '<option value="'. $t. '">'. $t. '</option>';

    }

    // Finish form and add footer

    echo '</select>';
    echo '<input type="submit" value="Select content to
    edit">';
    echo '</form>';
    echo '</div></div>';
    echo '<div id="licence">';
    echo '<a href="licence.txt" title="Copyright and licence
    details">Copyright &copy; 2013 Rob Somerville me@
    merville.co.uk</a>';
    echo '</div>';

}

function build_page_2($t,$sql,$skiplist){

    // HTML form

    echo '<div id="content">';
    echo '<div id="php">';
    echo '<div id="h1">2: Edit <?php echo $t;
    ?>&nbsp;content</div>';
    echo '<form action="amendcontent.php" method="post">';

    // Get the schema for that particular table

    $s = $sql[1];
    $s = str_replace ( '---P0---' , $t , $s );

    $result = mysql_fetchrows($s);
    $divstart = '<div class="inputname">';

```

```

echo '<input type="hidden" name="update"
value="' . $t . '">';

foreach($result as $row){

    // Loop through each field and build the form
    fields depending on the field    // type

    $field = $row[1];
    $fieldtype = $row[4];

    if (!in_array($field, $skiplist)) {

        if($fieldtype == "varchar"){

            echo $divstart . ucfirst($field) . '</
div><input class="varchar"
type="text" name="' . $field . '"><br />';

        }elseif($fieldtype == "int"){

            echo $divstart . ucfirst($field) . '</
div><input class="int"
type="text" name="' . $field . '"><br />';

        }elseif($fieldtype == "text"){

            echo $divstart . ucfirst($field) . '</
div><textarea rows="10" cols="30"
class="textarea" name="' . $field . '"></
textarea><br />';

        }else{

            // Shouldn't get here

            echo 'Error field(' . $field . ') ' .
$row[2] . ' | ' . $row[3] . ' | ' . $row[4] . ' | ' .
$row[5] . ' <br />';

        }

    }

}

// Finish form and add footer

echo '</select>';

echo '<input type="submit" value="Save changes">';
echo '</form>';
echo '</div></div>';
echo '<div id="licence">';
echo '<a href="licence.txt" title="Copyright and licence
details">Copyright &copy; 2013 Rob Somerville me@
merville.co.uk</a>';
echo '</div>';

}

function build_page_3($post){

    // HTML

    echo '<div id="content">';
    echo '<div id="php">';
    echo '<div id="h1">3: Save content</div>';
    echo '<ul>';

    foreach($post as $key => $value){

        // Just loop through and dump out values - we need
        to validate before adding to DB

        echo '<li><b>' . $key . '</b>: ' . $value . '</li>';

    }

    // End of form

    echo '</ul><br />';
    echo '<a href="amendcontent.php">Return to add content</
a>';

    echo '</div></div>';
    echo '<div id="licence">';
    echo '<a href="licence.txt" title="Copyright and licence
details">Copyright &copy; 2013 Rob Somerville me@
merville.co.uk</a>';
    echo '</div>';

}

```


Replace the code in (Listing 1) with the code in (Listing 2) and modify preload.js as well as global.css to match (Listing 3) and (Listing 4). This will provide the menu as shown in (Figure 2 & 3).

Add jquery support for each menu: Listing 3. Add some additional CSS so that the individual menus line up: Listing 4.

Step 2 – Create the menus table

In MySQL, create the menus table (Listing 5). Populate with some basic menus (Listing 6).

Step 3 – Build the amendcontent page

The amendcontent page is a PHP script that allows the user to add new content to the CMS. As we have not validated

Listing 8. additions to global.css

```
#php {
    min-height: 640px;
    margin-top: 160px;
}

.varchar {
    background-color: #ced8f8;
    border: 1px solid #FFF;
}

.int {
    background-color: #cef8f5;
    border: 1px solid #FFF;
}

.textarea {
    background-color: #e3f3dc;
    border: 1px solid #FFF;
}

.inputname {
    color: tomato;
    font-size: 12px;
    width: 100px;
    float: left;
    font-weight: bold;
}
```

Figure 5. Add your data

Figure 6. What will be saved

Useful links

- JQuery UI source – <http://jqueryui.com/resources/download/jquery-ui-1.10.3.zip>
- JQuery menu reference – <http://jqueryui.com/menu>
- PHP manual – <http://php.net/manual>

ed the user input yet, we'll just capture the input for now. Create a new PHP file called amendcontent.php in the root directory where index.php is already saved (Listing 7).

We need to add a small modification to global.css to line up the fields (Listing 8). Now visit <http://yoursite/amend-content.php> and you will have a dynamic form ready to save data to any table in the CMS. See (Figure 4-6).

In the next article

We will use the data from the menu tables to populate the JQuery menus and write some validation code for the user input prior to saving to the database.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.



Figure 4. Select the table to edit

FreeBSD Programming Primer – Part 9

In the ninth part of our series on programming, we will add some security to our CMS and refine our interface for adding content.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

In part 8 of the series, we created the PHP script `amendcontentpage.php` which allowed the user to add data to any table in the CMS (Figure 1 & 2). We will refine this page, and add a login page and the corresponding database table. Create a login table in MySQL to hold the user credentials (Listing 1).

We require a blob field as we will be storing binary rather than string data for the encrypted password. The auth field will be used to define the user rights later on, but for the moment setting a value of 1 via the form we will construct will be sufficient.

Now add the following to our `global.css` file to format the output from our `amendcontent.php` page (Listing 2).

Create a file in the `includes` directory called `login.inc` (Listing 3). This holds the name and secret key for the login cookie.

Create a new file `login.php` in the root directory of the application (Listing 4).

Finally, amend `amendcontent.php` as follows. As there are a lot of changes throughout the file, the script is detailed here in its entirety (Listing 5).

Hopefully, most of the code should be self explanatory, but here is a breakdown of the major functionality of each page.

Login.php

As we are storing the hashed value of the password in the database, rather than a text string that we can com-



2: Edit content

Group	Group name
Menu title	Menu title
Titleurl	http://www.google.com
Submenutitle	
Submenutitleurl	
Order	12
Enabled	1

Save changes

Figure 1. Select the table to edit



3: Save content

- update: menus
- group: Group name
- menu title: Menu title
- titleurl: <http://www.google.com>
- submenutitle:
- submenutitleurl:
- order: 12
- enabled: 1

[Return to add content](#)

Figure 2. Saving the data

Listing 1. *Create a login table in MySQL*

```
CREATE TABLE `login` (
  `id` int(5) unsigned zerofill NOT NULL AUTO_INCREMENT,
  `username` varchar(25) NOT NULL,
  `password` blob NOT NULL,
  `auth` int(1) NOT NULL,
  `timestamp` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=latin1;
```

Listing 2. *Additions to global.css*

```
.tablehdr {
  background-color: rosybrown;
  color: white;
  float: left;
  font-size: 14px;
  font-weight: bold;
  line-height: 25px;
  padding: 10px;
  width: 22%;
}

.tablerow1 {
  background-color: thistle;
  float: left;
  font-size: 14px;
  line-height: 25px;
  padding: 10px;
  width: 22%;
}

.tablerow2 {
  background-color: oldlace;
  float: left;
  font-size: 14px;
  line-height: 25px;
  padding: 10px;
  width: 22%;
}
```

Listing 3. *login.inc*

```
<?php

define('KEYNAME','gp19867fghlls');
define('LOGINKEY','117hkj23230rT');
```

Listing 4. *login.php*

```
<?php

require_once 'includes/cms.inc';
```

```
require INCLUDES . 'content.inc';
require INCLUDES . 'core.inc';
require INCLUDES . 'html.inc';
require INCLUDES . 'mysql.inc';
require INCLUDES . 'login.inc';

// SQL statements

$sql[0] = "SELECT password,auth FROM login
          WHERE username = '---P0---'
          AND password = '---P1---'";
$sql[1] = "INSERT INTO `login` (`username`, `password`,
          `auth`, `timestamp`)
          VALUES ('---P0---', ('---P1---'), '---P2---',
          now());";

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////

// Delete these 3 lines after first user added

if(!isset($_POST["action"])){
  createnewlogin();
}

////////////////////////////////////

// Page control logic

if(isset($_POST["action"])){

  $action = $_POST["action"];

  if($action == "validatelogin"){

    if(isset($_POST["username"]) && isset($_
```

```

POST["password"]){
    $username = $_POST["username"];
    $password = $_POST["password"];

    // We have valid credentials, validate

    validatelogin($username,
$password,$sql);
}

}elseif($action == "createnewlogin"){

    if(!isset($_COOKIE[KEYNAME])){

        // Create a new login to the system

        createnewlogin($username, $password,$sql);

    }else{

        // User failed cookie test, request them to
login

        requestlogindetails();
    }

}elseif($action == "appendnewlogin"){

    $username = $_POST["username"];
    $password = $_POST["password"];
    $auth = $_POST["auth"];

    appendnewlogin($username,$password,$auth,$sql);

}else{

    // Invalid action - request login details

    requestlogindetails();
}

}else{

    // First visit to page

    requestlogindetails();

}

}

function validatelogin($username, $password, $sql){

    // As the password is hashed and hopefully cannot be
decrypted,
    // We need to usend the encrypted password

    $hashed_password = hash('whirlpool', $password);

    // Fetch credentials from DB, if match create a login
cookie

    $s = $sql[0];
    $s = str_replace ( '---P0---' , $username , $s );
    $s = str_replace ( '---P1---' , $hashed_password , $s
);

    $result = mysql_fetchrows($s);

    foreach($result as $row){

        $auth = $row[1];

    }

    if ($auth == 1) {

        // Create auth cookie

        setcookie(KEYNAME, LOGINKEY, time()+3600, "/");

        // Display options

        $title = 'Welcome ` . $username;

        buildheader($title);
        echo wraptag("h1",$title);
        echo ahref('Add or amend content', '/amendcontent.
php');
        buildfooter();

    }else{

        // Try again

```


Are you in RED?

Meet iBLISS and turn into blue.

**Professional services and solutions - Imperva, McAfee, HP, Tenable.
Penetration tests, Application Security, Managed Security Services (MSS).**

Do as largest companies in Brazil, contact us!

www.ibliss.com.br info@ibliss.com.br +55 11 3255-3926



iBLISS
SEGURANÇA & INTELIGÊNCIA


```

        requestlogindetails();
    }
}

function createnewlogin(){
    $title = "Create new user";
    $class = "formcontrol";

    buildheader($title);
    echo wraptag("h1", $title);

    echo '<form action="login.php" method="post">';
    echo 'Username' . div('<input type="text"
name="username">', $class);
    echo 'Password' . div('<input type="password"
name="password">', $class);
    echo 'Auth' . div('<input type="text"
name="auth">', $class);
    echo '<input type="submit" value="Submit">';
    echo '<input type="hidden" name="action"
value="appendnewlogin">';
    echo '</form>';

    buildfooter();
}

function appendnewlogin($username, $password, $auth, $sql){
    // Create a new entry in the login table

    $hashed_password = hash('whirlpool', $password);

    $s = $sql[1];
    $s = str_replace ( '---P0---' , $username , $s );
    $s = str_replace ( '---P1---' , $hashed_password , $s
    );
    $s = str_replace ( '---P2---' , $auth , $s );

    mysql_select($s);
    requestlogindetails();
}

function requestlogindetails(){
    $title = "Please login";
    $class = "forminput";

    buildheader($title);

    echo wraptag("h1", $title);
    echo '<form action="login.php" method="post">';
    echo 'Username' . div('<input type="text"
name="username">', $class);
    echo 'Password' . div('<input type="password"
name="password">', $class);
    echo '<input type="submit" value="Submit">';
    echo '<input type="hidden" name="action"
value="validatelogin">';
    echo '</form>';

    buildfooter();
}

function buildheader($title){
    // As cookies need to be set before any output is
    sent to the browser
    // use a function call to build the page header

    // Build the page up to the body tag

    outfile(TEMPLATES . 'header.inc');

    echo wraptag('title', $title);
    echo HEAD;
    echo BODY;
    echo '<div id="content">';
    echo '<div id="php">';
}

function buildfooter(){
    echo '</div>';
    echo '</div>';
    echo '<div id="licence">';
    echo '<a href="licence.txt" title="Copyright and licence
details">Copyright &copy; 2013 Rob Somerville me@
merville.co.uk</a>';
    echo '</div>';
}

```

Listing 5. *amendcontent.php*

```

<?php

require_once 'includes/cms.inc';
require INCLUDES . 'content.inc';
require INCLUDES . 'core.inc';
require INCLUDES . 'html.inc';
require INCLUDES . 'mysql.inc';

// SQL statements

$sql[0] = "SELECT COUNT(DISTINCT TABLE_NAME) FROM
INFORMATION_SCHEMA.COLUMNS
WHERE table_schema = 'freebsdcms'
AND TABLE_NAME = '---P0---'";
$sql[1] = "SELECT TABLE_NAME,COLUMN_NAME,COLUMN_
DEFAULT,IS_NULLABLE,DATA_TYPE,CHARACTER_MAXIMUM_
LENGTH
FROM INFORMATION_SCHEMA.COLUMNS
WHERE table_schema = 'freebsdcms'
AND TABLE_NAME = '---P0---'
ORDER BY table_name, ordinal_position";
$sql[2] = "SELECT * FROM ---P0--- ORDER BY id DESC";
$sql[3] = "SELECT COLUMN_NAME FROM INFORMATION_SCHEMA.
COLUMNS WHERE TABLE_SCHEMA = 'freebsdcms' AND TABLE_
NAME = '---P0---'";
$sql[4] = "SELECT `---P0---` FROM ---P1--- WHERE id ='--
-P2---'";

// The tables we will allow the user to edit via this
form

$tables[] = "faqs";
$tables[] = "menus";
$tables[] = "news";
$tables[] = "pages";

// Fields that are automatically assigned via a default
value in MySQL table definition

$skiplist[] = "id";
$skiplist[] = "timestamp";

////////////////////////////////////////
////////////////////////////////////////
////////////////////////////////////////

// Build the page up to the body tag

outfile(TEMPLATES . 'header.inc');

echo wraptag('title', 'Content Input');
echo HEAD;
echo BODY;

// Page control logic

if(isset($_POST["table"])){

    // User has not selected a table or we are testing
    their result

    $t = $_POST["table"];

    if (!in_array($t, $tables)) {

        // If the table is not on allowed list, bail to
        the first page

        build_page_1($tables);

    }else{

        // Check selected table is valid

        $s = $sql[0];

        // Replace the marker in the SQL statement with
        the chosen value

        $s = str_replace ( '---P0---' , $t , $s );

        $result = mysql_select($s);
        $valid_table_count = $result['COUNT(DISTINCT
        TABLE_NAME)'];

        if($valid_table_count == 1){

            // Valid table selected - present form to edit data
            build_page_2($t,$sql,$skiplist);

        }else{

            // Send user to first page
            build_page_1($tables);

        }

    }

}

```

```

}elseif(isset($_POST["update"])){

    // Save the input. As we have not validated this,
    just display for now

    build_page_3($_POST);
}elseif(isset($_POST["id"])){

    // Save the input. As we have not validated this,
    just display for now

    echo "Update old record";
}else{

    // Invalid value - return to start

    build_page_1($tables);
}

////////////////////////////////////
////////////////////////////////////
////////////////////////////////////

function build_page_1($tables){

    // HTML form definition

    echo '<div id="content">';
    echo '<div id="php">';
    echo '<div id="h1">Select content</div>';
    echo '<form action="amendcontent.php" method="post">';

    // Build the list of tables

    $stablecontrol = '';
    $stablecontrol .= '<select name="table">';

    foreach ($tables as $t){

        // $tables is an array - split each value out

        $stablecontrol .= '<option value="' . $t . '>' . $t . '</option>';

    }

    $stablecontrol .= '</select>';

    // Build the edit options

    $editcontrol = '';
    $editcontrol .= '<input type="radio" name="inputmode"
        value="new" checked="checked">Add new content';
    $editcontrol .= '<input type="radio" name="inputmode"
        value="update">Update current content';

    // Build the submit option

    $submitcontrol = '';
    $submitcontrol .= '<input type="submit" value="Create
        content">';

    // Add the DIV to format the controls

    echo div($stablecontrol, 'formcontrol');
    echo div($editcontrol, 'formcontrol');
    echo div($submitcontrol, 'formcontrol');

    // Complete the form

    echo '</form>';
    echo '</div></div>';
    echo '<div id="licence">';
    echo '<a href="licence.txt" title="Copyright and licence
        details">Copyright &copy; 2013 Rob Somerville me@
        merville.co.uk</a>';
    echo '</div>';
}

function build_page_2($t,$sql,$skiplist){

    // HTML form

    echo '<div id="content">';
    echo '<div id="php">';

    // Check we have a valid inputmode

    if(!isset($_POST["inputmode"])){

        echo "Error: Invalid inputmode";
        exit;

    }else{

```



Dr.Web 9.0

for Windows —
the rapid response anti-virus

1. Reliable protection against the threats of tomorrow
2. Reliable protection against data loss
3. Secure communication, data transfer and Internet search



```

if($_POST["inputmode"] == "new" || (isset($_
POST["rowid"]))) {

    // New content - populate with selected value if we
    have arrived here
    // via an update content request.

    if(isset($_POST["rowid"])) {

        $rowid = $_POST["rowid"];
        $populate = TRUE;

    }else{

        $rowid = "";
        $populate = FALSE;

    }

    echo '<div id="h1">Create new `.$t.` content</
div>';
    echo '<form action="amendcontent.php"
method="post">';

    // Get the schema for that particular table

    $s = $sql[1];
    $s = str_replace ( '---P0---' , $t , $s );

    $result = mysql_fetchrows($s);
    $divstart = '<div class="inputname">';
    $action = 'Save';

    echo '<input type="hidden" name="update"
value=""'.$t.'">';

    foreach($result as $row) {

        // Loop through each field and build the form fields
        depending on the field type

        $field = $row[1];
        $fieldtype = $row[4];

        if (!in_array($field, $skiplist)) {

            if($fieldtype == "varchar") {

                $value = populatefields($sql[4], $field, $t, $row
id, $populate);

                echo $divstart . ucfirst($field).'</div><input
class="varchar" type="text" name=""'.$field.'"
value=""'.$value.'"><br />';

            }elseif($fieldtype == "int") {

                $value = populatefields($sql[4], $field, $t, $ro
wid, $populate);

                echo $divstart . ucfirst($field).'</div><input
class="int" type="text" name=""'.$field.'"
value=""'.$value.'"><br />';

            }elseif($fieldtype == "text") {

                $value = populatefields($sql[4], $field, $t,
$rowid, $populate);

                echo $divstart . ucfirst($field).'</
div><textarea rows="10" cols="30" class="textarea"
name=""'.$field.'">'.$value.'</textarea><br />';

            }else{

                // Shouldn't get here

                echo 'Error field(`.$field.`) `.`
$row[2].`|`.`.$row[3].`|`.`.$row[4].`|`.`.$row[5].`<br
/>';

            }

        }

    }

    //echo '</select>';

}elseif($_POST["inputmode"] == "update") {

    echo '<div id="h1">Select content `.$t.`</div>';
    echo '<form action="amendcontent.php"
method="post">';
    echo '<input type="hidden" name="table"
value=""'.$t.'">';

```



```

echo '<input type="hidden" name="inputmode"
value="new">';

$s = $sql[3];

// Replace the marker in the SQL statement with the
chosen value

$s = str_replace ( '---P0---' , $t , $s );

if($t == 'menus'){

    // DB schema is different
    // NB: Maximum cols = 3 unless mods to CSS
    performed

    $displaycols = array(2, 4, 5);

}else{

    // Everything else

    $displaycols = array(1, 2, 3);

}

// Get the field names for our table

$titles = mysql_fetchrows($s);

// Build the title row

echo div('Select', 'tablehdr');

foreach ($displaycols as $offset) {

    echo div($titles[$offset][0], 'tablehdr');

}

$s = $sql[2];
$zebra = 0;
$action = 'Update';

// Replace the marker in the SQL statement with the
chosen value
$s = str_replace ( '---P0---' , $t , $s );

$result = mysql_fetchrows($s);

foreach($result as $row){

    if($zebra == 0){

        $class = 'tablerow1';
        $zebra = 1;

    }elseif($zebra == 1){

        $class = 'tablerow2';
        $zebra = 0;

    }

    // Radio button control

    $editcontrol = '<input type="radio" name="rowid"
value="" . $row[0] . ">';

    // Check formatting and output

    formatcontentedit($row, $class, $displaycols,
    $editcontrol);

}

}else{

    echo "Error: Invalid inputmode";
    exit;

}

}

// Finish form and add footer

echo '<input type="submit" value="" . $action . ' . $t . '
item">';
echo '</form>';
echo '</div></div>';
echo '<div id="licence">';
echo '<a href="licence.txt" title="Copyright and licence
details">Copyright &copy; 2013 Rob Somerville me@
merville.co.uk</a>';
echo '</div>';
}

```

```

function build_page_3($post){

    // HTML

    echo '<div id="content">';
    echo '<div id="php">';
    echo '<div id="h1">Save content</div>';
    echo '<ul>';

    foreach($post as $key => $value){

        // Just dump out values - we need to validate before
        // adding to DB

        echo '<li><b>'. $key. '</b>: '. $value. '</li>';

    }

    // End of form

    echo '</ul><br />';
    echo '<a href="amendcontent.php">Return to add content</a>';

    echo '</div></div>';
    echo '<div id="licence">';
    echo '<a href="licence.txt" title="Copyright and licence
    details">Copyright &copy; 2013 Rob Somerville me@
    merville.co.uk</a>';
    echo '</div>';

}

function formatcontentedit($row, $class, $displaycols,
    $editcontrol){

    // Formats the rows from our select query in zebra
    // format.
    // To prevent the CSS from breaking due to NULL
    // content
    // and displays the appropriate rows as the menu
    // schema is
    // different from everything else.

    // Display the radio button

    echo div($editcontrol, $class);

    // Format each row

    foreach ($displaycols as $offset) {

        // First check we have some content - use a NBSP
        // if NULL or blank

        if($row[$offset] == ''){

            $row[$offset] = '&nbsp;';

        }

        // Ensure length < 25 chars, else add ellipses

        if(strlen($row[$offset]) > 25){

            $row[$offset] = substr($row[$offset],0,24) .
            ' ...';

        }

        // Display each field from the row

        echo div($row[$offset], $class);

    }

}

function populatefields($sql,$field,$t,$rowid,$populate){

    if($populate){

        $s = str_replace ( '---P0---' , $field , $sql );
        $s = str_replace ( '---P1---' , $t , $s );
        $s = str_replace ( '---P2---' , $rowid , $s );

        $v = mysql_fetchrows($s);

        $value = $v[0][0];

    }else{

        $value = "";

    }

    return $value;

}

```



FreeBSD®


Create new user

Username
admin

Password

Auth
1

Figure 3. Creating a new user



FreeBSD®

Select content

☒ Add new content ☐ Update current content

Figure 6. Choose your content type add new or update



FreeBSD®

Please login

Username
Password

Figure 4. Logging in




FreeBSD®

Create new faqs content

Title
Heading
Content

Status

Figure 7. Adding a new FAQ



FreeBSD®

Welcome admin

[Add or amend content](#)

Figure 5. Add or amend content



FreeBSD®

Select content faqs

Select	title	heading	content
<input type="radio"/>	FAQ 10	Tenth FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 9	Ninth FAQ	Aenean volutpat, ligula ...
<input checked="" type="radio"/>	FAQ 8	Eighth FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 7	Seventh FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 6	Sixth FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 5	Fifth FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 4	Fourth FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 3	Third FAQ	Lorem ipsum dolor sit am ...
<input type="radio"/>	FAQ 2	Second FAQ	Aenean volutpat, ligula ...
<input type="radio"/>	FAQ 1	First FAQ	Aenean volutpat, ligula ...

Figure 8. Choosing an existing FAQ to edit

pare, we need to initially seed the database with a valid username and password the first time login.php is run. Once the login table is updated, the call to `createnewlogin()` can be removed. The page control logic branches depending on the action we want to achieve, and the corresponding function calls either build an HTML form, query the database or add a user to the database.

Amendcontent.php

Most of the action takes place within `build_page_2`. In the previous version, we could not select any previously entered content so to add this functionality we have added an intermediate step which displays all the content avail-

able to be edited. Once the user selects the table record to be amended, this is fed back into our original form, which is essentially identical whether we are updating or adding content. Some “bells and whistles” are added in the form of zebra striping of the table rows, and the automatic generation of the titles. As the script is referring directly to the database, we need a conditional branch at line 266 as the menu table has a different schema from the pages, news and FAQ content.

Getting it to work

Visit the login item via the menu and create a new user and password with an auth level of 1. Check the user has been

FreeBSD®

Create new faqs content

Title: FAQ 8

Heading: Eighth FAQ

Content: Aenean volutpat, ligula vitae laoreet dapibus

Status: 2

Save faqs item

Copyright © 2013 Bob Searcote, net@searcode.co.uk

Figure 9. Editing a pre-existing FAQ

FreeBSD®

Select content menus

Select	menu title	sub-menu title	sub-menu href
<input type="radio"/>	Pages	Page 3	/page/3
<input checked="" type="radio"/>	Pages	Page 2	/page/2
<input type="radio"/>	Pages	Page 1	/page/1
<input type="radio"/>	Pages		
<input type="radio"/>	Home		

Update menus item

Copyright © 2013 Bob Searcote, net@searcode.co.uk

Figure 11. Picking an existing menu to edit

FreeBSD®

Save content

- update: faqs
- title: FAQ 8
- heading: Eighth FAQ
- content: Aenean volutpat, ligula vitae laoreet dapibus
- status: 2

Return to add content

Copyright © 2013 Bob Searcote, net@searcode.co.uk

Figure 10. Saving the FAQ

FreeBSD®

Create new menus content

Group: jquerymenu

Menu title: Pages

Title url:

Submenu title: Page 2

Submenu href: /page/2

Order: 2

Enabled: 1

Save menus item

Copyright © 2013 Bob Searcote, net@searcode.co.uk

Figure 12. Saving a menu item

Useful links

PHP manual – <http://php.net/manual>

added to the login table and remove the 3 lines from the start of login.php as commented. Revisit login.php, login and proceed to edit your content as desired (Figure 3-12).

Further tuning

The security is poor – we can have multiple users with the same name and password. A better form of encryption other than hashing is desirable, and we are missing lots of backlinks etc. We should also refactor the code e.g. the license and footers.

In the next part

We will address these issues and more.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.



FreeBSD



[GEEKED AT BIRTH]



You can talk the talk.
Can you walk the walk?

[IT'S IN YOUR DNA]

LEARN:

Advancing Computer Science
Artificial Life Programming
Digital Media
Digital Video
Enterprise Software Development
Game Art and Animation
Game Design
Game Programming
Human-Computer Interaction
Network Engineering
Network Security
Open Source Technologies
Robotics and Embedded Systems
Serious Game and Simulation
Strategic Technology Development
Technology Forensics
Technology Product Design
Technology Studies
Virtual Modeling and Design
Web and Social Media Technologies

www.uat.edu > 877.UAT.GEEK

FreeBSD Programming Primer – Part 10

In the tenth part of our series on programming, we will improve the login process, add more security, and keep spam robots under control.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

In the previous article we put in place a very crude login system that allowed anyone to login to our CMS and add content. We assume that the user has been correctly authenticated by comparing their password against a hashed

password stored in the CMS database, then writing a cookie at the client side. It is then a simple matter of checking that authorization has been granted prior to carrying out sensitive actions (e.g. adding a user or amending content).

Listing 1. Logout function

```
function logout() {

    setcookie(KEYNAME, LOGINKEY, time()-3600, "/");
    echo "You have been logged out";

}
```

Listing 2. Adding the logout logic

```
}elseif($action == "appendnewlogin"){

    $username = $_POST["username"];
    $password = $_POST["password"];
    $auth = $_POST["auth"];

    appendnewlogin($username,$password,$auth,$sql);

}elseif($action == "logout"){

    // Logout the user

    logout();

}else{

    // Invalid action - request login details
```

```
requestlogindetails();

}
```

Listing 3. logoutform

```
function logoutform() {

    // Check if user is logged in, if so display the logout
    button.

    require_once 'includes/cms.inc';
    require INCLUDES . 'login.inc';

    if(isset($_COOKIE[KEYNAME])){

        echo '<div id="logout">';
        echo '<form action="login.php" method="post">';
        echo '<input type="submit" value="logout">';
        echo '<input type="hidden" name="action"
        value="logout">';
        echo '</form>';
        echo '</div>';

    }

}
```

Unfortunately, exposing any login system on the World Wide Web leaves us open to undesirable elements. Brute force attacks (repeatedly attempting a login using dictionary attacks) and spambots that want to add advertising or phishing spam are commonplace, and our basic login system needs to defend against this. We also need to add logout functionality to every page that requires it.

The logout functionality

As the parameters passed to the cookie that is set when we are logged in, it makes sense to hold the logout func-

tion as part of the *login.php* page. We can then detect a logout post event to *login.php* and delete the cookie by setting the expire date to a time in the past. Add the following code at the end of *login.php* (Listing 1).

Now we need to check for a post event that carries the value logout. Add the following elseif branch between *append* and the closing *else* (Listing 2).

We now need a *logoutform()* function that will provide a logout button whenever a user is logged in to the system. If we check whether or not the user is logged in we can place this in the footer of all pages where login / logout



Figure 1. Login – no cookie present



Figure 2. Cookie present but no logout button

Listing 4. Test to see if user is logged in

```
function ifnotloggedin(){
    // Check if user is logged in, if not, redirect to
    // login form

    require_once 'includes/login.inc';

    if(!isset($_COOKIE[KEYNAME])){

        header( 'Location: http://'.CMSDOMAIN.'/login.php' );
    }
}
```

Listing 5. Set our domain

```
// Our domain

define("CMSDOMAIN", '192.168.0.118');
```

Listing 6. amendcontent.php

```
// Check we are logged in
ifnotloggedin();
// Build the page up to the body tag
```

```
outfile(TEMPLATES . 'header.inc');
```

Listing 7. phpinfo.php

```
<?php

// Check we are logged in

require_once 'includes/cms.inc';
require INCLUDES . 'content.inc';
require INCLUDES . 'core.inc';
ifnotloggedin();
phpinfo();
logoutform();
```

Listing 8. amendcontent.php and login.php

```
echo BODY;
logoutform();
```

Listing 9. global.css

```
#logout {
    float: right;
    background-color: tomato;
    padding: 5px;
    border-radius: 10px;
}
```

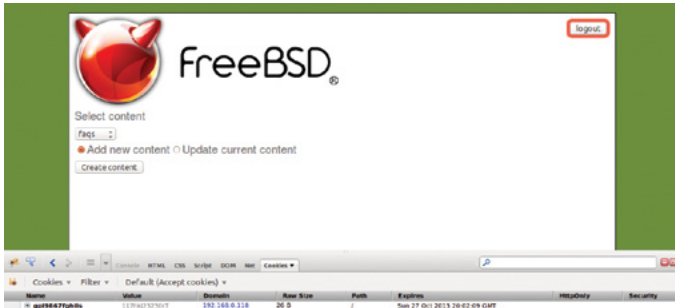


Figure 3. Cookie present – login button visible

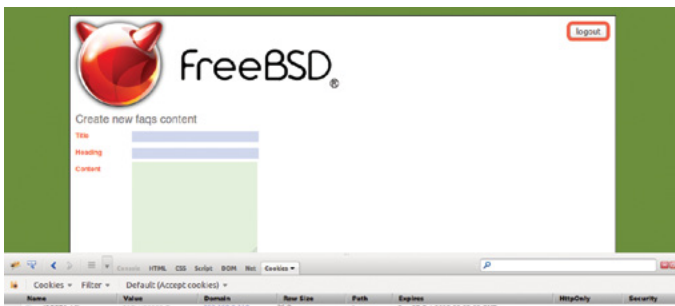


Figure 4. Logout button visible on new faq's page

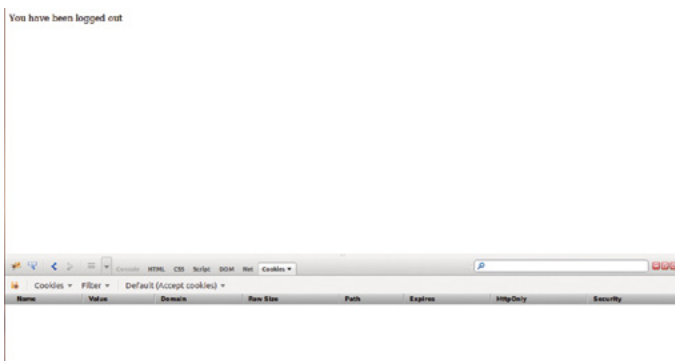


Figure 5. Logout message

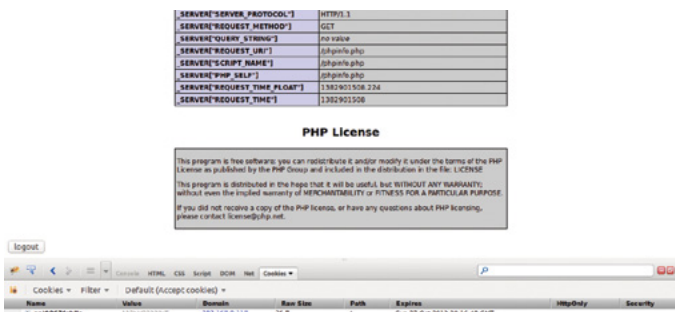


Figure 6. Logout button on phpinfo.php

Listing 10. validatelogin()

```
setcookie(KEYNAME, LOGINKEY, time()+3600, "/");

// Display options

$title = 'Welcome ' . $username;

buildheader($title);
echo wraptag("h1", $title);

echo ahref('Add or amend content', '/amendcontent.
    php');

buildfooter();
```

Listing 11. Replacement buildheader();

```
function buildheader($title, $forcelogout = 0){

    // As cookies need to be set before any output is
    // sent to the browser
    // use a function call to build the page header

    // Build the page up to the body tag

    outfile(TEMPLATES . 'header.inc');

    echo wraptag('title', $title);
    echo HEAD;
    echo BODY;
    logoutform($forcelogout);

    echo '<div id="content">';
    echo '<div id="php">';

}
```

Listing 12. Amended validatelogin();

```
setcookie(KEYNAME, LOGINKEY, time()+3600, "/");

// Display options

$title = 'Welcome ' . $username;

buildheader($title, 1);
echo wraptag("h1", $title);
```

functionality is required, and the button will be displayed only if the user is logged in. Add this to the end of `core.inc` (Listing 3).

We need to add a function call to check if the user is logged in or not, and redirect them to the login page if they are not. Add this at the end of `core.inc` (Listing 4).

As we cannot guarantee that the user does not spoof HTTP headers for the redirect, define our CMS Domain in `cms.inc`. Replace 192.168.0.118 with either the IP address or domain name of your server (if accessible via DNS). (Listing 5)

Add the `ifnotloggedin()` function call to the beginning of `amendcontent.php` and replace `phpinfo.php` with the content in Listing 7 (Listing 6 & 7).



Figure 7. The fixed welcome page

Listing 13. Amended `logoutform()`:

```
function logoutform($forcelogout = 0){

    // Check if user is logged in, if so display the
    // logout button.

    require_once 'includes/login.inc';

    if(isset($_COOKIE[KEYNAME]) || $forcelogout == 1){

        echo '<div id="logout">';
```

Listing 14. Add `spambot` field to `requestlogindetails()` in `login.php`

```
echo 'Username' . div('<input type="text"
    name="username">', $class);
echo 'Password' . div('<input type="password"
    name="password">', $class);
echo 'Email' . div('<input type="text"
    name="email">', 'loginemail');
echo '<input type="submit" value="Submit">';
```

Listing 15. Remove the comment out from `createnewlogin`, suffix with `//` to revert to normal login action

```
if(!isset($_POST["action"])){

    createnewlogin();

}
```

Listing 16.

```
CREATE TABLE `access` (
    `id` int(10) unsigned zerofill NOT NULL AUTO_INCREMENT,
```

```
`ipaddress` varchar(64) NOT NULL,
`page` varchar(64) NOT NULL,
`status` int(1) NOT NULL,
`timestamp` timestamp NOT NULL DEFAULT CURRENT_
    TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=0 DEFAULT CHARSET=latin1;
```

Listing 17. `sqlstatements.inc`

```
<?php
/*
 *
 * sqlstatements.inc
 * Contains CMS SQL statements
 *
 */

$sql[0] = "INSERT INTO `access` (`ipaddress`, `page`,
    `status`, `timestamp`)
    VALUES ('---P0---', ('---P1---'), '---P2---',
    now());";

$sql[1] = "SELECT status FROM access
    WHERE ipaddress = '---P0---'
    AND status > 0
    LIMIT 1";
```

Add the following line to `cms.inc` [Listing 7]

```
// Honeypot for bad traffic

define("HONEYPOT", 'www.google.com');
```

Listing 18.*mysql_select()*

```
function mysql_select($sql) {

    $db = new mysqli(DBSERVER, DBUSER, DBPASSWORD,
CMSDB);

    if ($db->connect_errno > 0) {
        die('Unable to connect to database [' .
$db->connect_error . ']');
    }

    if (!$result = $db->query($sql)) {
        if (DEBUG) {
            die('There was an error running the query ['
. $db->error . ']');
        } else {
            die('');
        }
    }

    // Pass our results to an array to be returned

    if(isset($result->num_rows)){

        $r = array();

        $r[] = $result->num_rows;    // No of rows returned
        $r[] = $db->field_count;    // No of columns in
table
        $r[] = $db->affected_rows;  // No of rows affected
e.g. update / delete

    // Append the results to our result count

    if ($result->num_rows != 0) {

        $r = array_merge($r, $result->fetch_
array(MYSQLI_ASSOC));
    }

    // Free the result

    $result->free();

}else{

    $r = NULL;
```

```
    }

    // Close the connection

    $db->close();

    return $r;
}
```

Listing 19.*Additions to core.inc*

```
function loginsecurity(){

    require INCLUDES . 'sqlstatements.inc';

    // Get client IP address

    $ip = $_SERVER["REMOTE_ADDR"];

    if(isset($_POST["email"])){

        // email will always be set, check if it is populated

        if($_POST["email"] != ''){

            // Ban 'em

            banip($ip, 'login.php');

        }

    }else{

        // Check that they have not been flagged as
suspicious

        $s = $sql[1];
        $s = str_replace ( '---P0---' , $ip , $s );

        $result = mysql_fetchrows($s);

        if($result){

            foreach($result as $row){

                $status = $row[0];

            }

        }else{
```



```

    $status = 0;

}

// Redirect to our honeypot if status is set

if($status != 0){

    header( 'Location: http://' . HONEYPOT ) ;

}

}

function banip($ip, $page){

    require INCLUDES . 'sqlstatements.inc';

    // Add to our banlist

    $s = $sql[0];
    $s = str_replace ( '---P0---' , $ip , $s );
    $s = str_replace ( '---P1---' , $page , $s );
    $s = str_replace ( '---P2---' , 1 , $s );

    mysql_select($s);

    // Redirect to our honeypot

    header( 'Location: http://' . HONEYPOT ) ;

}

function logip($page){

    require INCLUDES . 'sqlstatements.inc';

    // Just log a visit

    $ip = $_SERVER["REMOTE_ADDR"];

    $s = $sql[0];
    $s = str_replace ( '---P0---' , $ip , $s );
    $s = str_replace ( '---P1---' , $page , $s );
    $s = str_replace ( '---P2---' , 0 , $s );

```

```
mysql_select($s);
```

```
}
```

Listing 20. Replacement validatelogin()

```

function validatelogin($username, $password, $sql){

    // Create a session to keep track of our login
    attempts

    session_start();

    // As the password is hashed and hopefully cannot be
    decrypted,
    // We need to send the encrypted password

    $hashed_password = hash('whirlpool', $password);

    // Fetch credentials from DB, if match create a login
    cookie

    $s = $sql[0];
    $s = str_replace ( '---P0---' , $username , $s );
    $s = str_replace ( '---P1---' , $hashed_password , $s
    );

    $result = mysql_fetchrows($s);

    if($result){

        foreach($result as $row){

            $auth = $row[1];

        }

    }else{

        $auth = 0;

    }

    if ($auth == 1) {

        // Log our successful login

        logip('login.php');
    }
}

```

```

    // Reset our attempt count in case they login
    again

    unset($_SESSION['loginattempts']);

    // Create auth cookie

    setcookie(KEYNAME, LOGINKEY, time()+3600, "/");

    // Display options

    $title = 'Welcome ' . $username;

    buildheader($title,1);
    echo wraptag("h1",$title);

    echo ahref('Add or amend content', '/amendcontent.
    php');

    buildfooter();

}

// Keep a track of the number of attempts we have
made at logging in

if(isset($_SESSION['loginattempts'])) {

    $_SESSION['loginattempts'] = $_
    SESSION['loginattempts'] + 1;

}

$_SESSION['loginattempts'] = 1;

}

// If they have exceeded our limit, ban 'em

if($_SESSION['loginattempts'] > 3){

    $ip = $_SERVER["REMOTE_ADDR"];

    banip($ip, 'login.php');

}

// Try again

```

```
requestlogindetails();
```

Listing 21. Modified buildheader()

```

// As cookies need to be set before any output is sent
to the browser
// use a function call to build the page header

// Check we are not on the ban list and that we are
not a spam robot

loginsecurity();

// Build the page up to the body tag

outfile(TEMPLATES . 'header.inc');

```

Listing 22. Hide the email address field

```

.loginemail {
    visibility: hidden !important;
}

```



Figure 8. The login page with the email “honeytrap”

Add the `logoutform()`; after every occurrence after `echo BODY`; in `login.php` and `amendcontent.php` (Listing 8).

Add the following to `global.css` to highlight and position the logout button (Listing 9).

With Firebug enabled in Firefox, check that a cookie called `gpl9867fgh11s` is created when a user is logged in. The logout button should appear on all pages except the second time `login.php` is called and we arrive at the welcome page (Figure 1 – 6).

Now this is a problem, as we should be able to logout immediately after we login. Subsequent calls to `login.php` will show the logout button. So what is happening here? The problem lies in the `validatelogin()` function (Listing 10).

We must set the cookie prior to creating the page header, but as the cookie data is generated at the client browser side when the page is loaded, as far as the PHP code running at the server side is concerned the cookie is not present yet. We can fool `buildheader()` by passing a parameter to force the display of the logout button (Listing 11 – 13). This will result in `login.php` working as desired (Figure 7).

Spambots and robots

While we could use the very effective Apache `MOD_SECURITY` module to trap bad behaviour, this can be tricky to set up. What we will do here is monitor behaviour in two ways. First, we will create a hidden field that a normal user will not see under normal circumstances, which most spam-robots will fill in assuming it is a genuine field. On completing the field, our CMS will automatically ban all connections from that IP address to `login.php` permanently.

We will also check that no more than 3 invalid attempts are made to the `login.php` page, and if that is exceeded, that IP address will be banned as well.

First create another testuser by changing `login.php` as follows and visit `login.php` anew to create another user (e.g. Test, Test, Auth = 1). Don't worry about the error messages – we will fix them later. Once you have created the new user, go back and comment out `createnewlogin()`; and check that you can login as the test user (Listing 14 and 15).

If you visit `login.php` you should be able to login as Test (Ignore the Email field), then Logout. (Figure 8). Now create our access table in MySQL to hold our banlist (Listing 16). Now create the file `sqlstatements.inc` in our includes directory (Listing 17). Replace the `mysql_select()` function call in `mysql.inc` with the following code (Listing 18). This fixes a bug where a PHP error is raised when no results are returned. Add the following function calls to `core.inc` (Listing 19). Replace `validatelogin()` in `login.php` with the following code (Listing 20). Modify `buildheader()` in `login.php` to call `loginsecurity()` (Listing 21).

Useful links

PHP manual – <http://php.net/manual>

Testing

It is recommended that you run Firebug to view the cookies and PHP sessions generated during this test. Clear all cookies etc. from your browser and visit `login.php`:

- Login as Test with the correct password. You should be able to login. Logout.
- Login as Test with the correct password and an email address. You should be redirected to `google.com`. Any visits to `login.php` will cause a redirect to `google.com`.
- Use Adminer to clear all the entries from the access table.
- Visit `login.php` and click on the submit button 3 times without making any input. You should be redirected on the 4th attempt.
- Use Adminer to clear all the entries from the access table.
- Visit `login.php` and login and logout as normal. Your access attempts should be logged correctly with IP address and date.
- Login with a mixture of bad username and good password, good username and bad password. You should be banned on your 4th login attempt.

CSS modification

Finally, add the following code to `global.css` and refresh your browser with Ctrl F5 a couple of times to clear the cache. The email field should now be invisible to human visitors, but available to robots etc. (Listing 22).

Next steps

It might be a good idea to add the banlist functionality to all pages on a failed login etc. and keep a tally of what pages are accessed etc. legitimately. We also need to add the facility to add a user rather than manually editing code each time. Our CMS is getting quite large, with over 2,100 lines of code (excluding the JQuery libraries) so we will look at refactoring some of this code in the next article.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

FreeBSD Programming Primer – Part 11

In the penultimate part of our series on programming, we will look at using the Netbeans Integrated Development Environment to debug and edit our CMS.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

Unfortunately, the Internet gremlins have got me at the moment so this how-to is going to be very short. My local telco is currently rolling out fibre in the area, and my ADSL internet connection is very unreliable, but hopefully I will be able to wrap up the programming primer in part 12 with a bumper edition.

While debugging at the command line using echo statements or commenting out code is possible, a more frequent scenario is that our project will be residing on a remote server and we will need to see the actual processes in action. Often developers will have a local copy of the LAMP stack on their PC or laptop, so that they can debug locally. However, what happens when our development environment is on a laptop and the code is on a remote server? A frequent approach is to use an Integrated Development Environment (IDE) with a built in file transfer utility. Coupled with Xdebug, which supports PHP, we can download our remote code and debug (step through) each line, examine variables etc. To do this, we will need to install Xdebug on our server and install the IDE of our choice on an available local PC. This can be FreeBSD, Windows or Linux, but in my case I was using an Ubuntu desktop. The IDE installation will vary from environment to environment, full details can be found at <https://netbeans.org>. The IP address of my desktop PC for this exercise was 192.168.0.123.

Installing Xdebug

Rather than using the FreeBSD provided software, I downloaded the latest version from <http://xdebug.org>. The reason for this is that in the past I have had prob-

lems getting the standard packaged version of Xdebug working with certain distro's, where as the latest Xdebug

Listing 1. Install Xdebug

```
tar -xvzf xdebug-2.2.3.tgz
cd xdebug-2.2.3
phpize
./configure --enable-xdebug
make
cd modules

cp xdebug.so /usr/local/lib/php/20100525/
touch /var/log/xdebug.log
chmod 666 /var/log/xdebug.log
touch /usr/local/etc/php/xdebug.ini
```

Listing 2. /usr/local/etc/php/xdebug.ini

```
zend_extension=/usr/local/lib/php/20100525/xdebug.so
xdebug.remote_enable=1
xdebug.remote_host="192.168.0.123"
xdebug.remote_port=9000
xdebug.remote_handler="dbgp"
xdebug.remote_mode=req
xdebug.profiler_enable = 1
xdebug.remote_log=/var/log/xdebug.log
```

Listing 3. Restarting Apache

```
/usr/local/etc/rc.d/apache22 stop
/usr/local/etc/rc.d/apache22 start
```

and latest Netbeans IDE always seem to work OK together. Once you have downloaded the latest version of the tarball (Currently xdebug-2.2.3.tgz) into your home directory, on the remote server (192.168.0.118) as root, perform the following (Listing 1).

Add the following to `/user/local/etc/php/xdebug.ini` (Listing 2).

Replace 192.168.0.123 with the IP address of your client machine.

Restart Apache (Listing 3).

If we now login as admin and visit our PHPinfo page at `http://192.168.0.118/phpinfo.php`, we should see that Xde-

Directive	Local Value	Master Value
xdebug.auto_trace	Off	Off
xdebug.cli_color	0	0
xdebug.collect_assignments	Off	Off
xdebug.collect_includes	On	On
xdebug.collect_params	0	0
xdebug.collect_return	Off	Off
xdebug.collect_vars	Off	Off
xdebug.coverage_enable	On	On
xdebug.default_enable	On	On
xdebug.dump.COOKIE	no value	no value
xdebug.dump.ENV	no value	no value
xdebug.dump.FILES	no value	no value
xdebug.dump.GET	no value	no value
xdebug.dump.POST	no value	no value
xdebug.dump.REQUEST	no value	no value
xdebug.dump.SERVER	no value	no value
xdebug.dump.SESSION	no value	no value
xdebug.dump.globals	On	On
xdebug.dump_once	On	On
xdebug.dump_undefined	Off	Off
xdebug.extended_info	On	On
xdebug.file_link_format	no value	no value
xdebug.idekey	no value	no value

Figure 1. PHP Xdebug installed

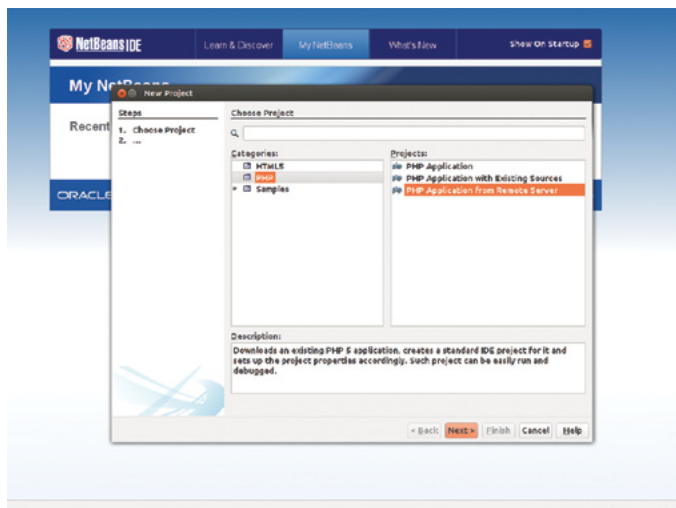


Figure 2. Create a new project with PHP application on remote server

bug is installed and running (Figure 1). If you have not already done so, download and install Netbeans on a local PC of your choice. You will need a working Java installation and Firefox installed for this to work.

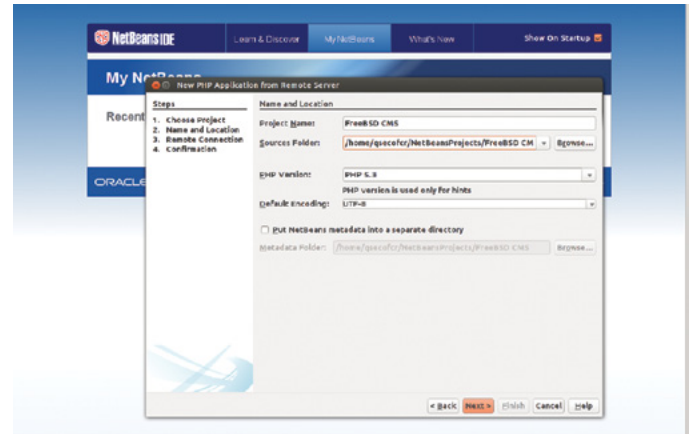


Figure 3. Give the project a name

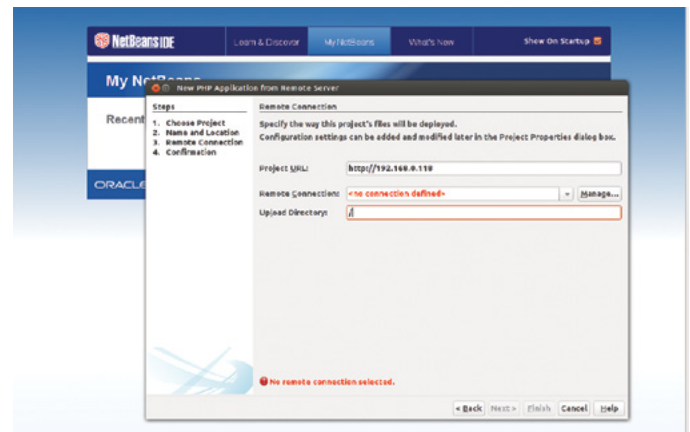


Figure 4. Create a new remote connection by clicking on Manage

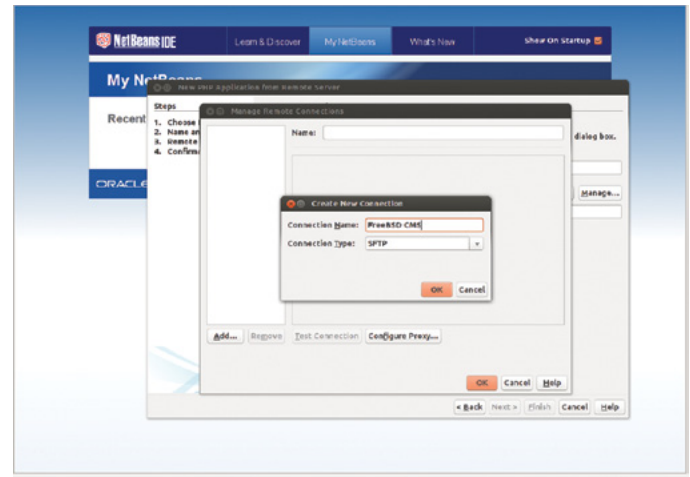


Figure 5. Create a new SFTP connection (SSH must be running on Port 22 of your server)

Useful links

- Xdebug: <http://xdebug.org>
- Netbeans: <http://php.net/manual>

- ## Useful links
- Xdebug: <http://xdebug.org>
 - Netbeans: <http://php.net/manual>

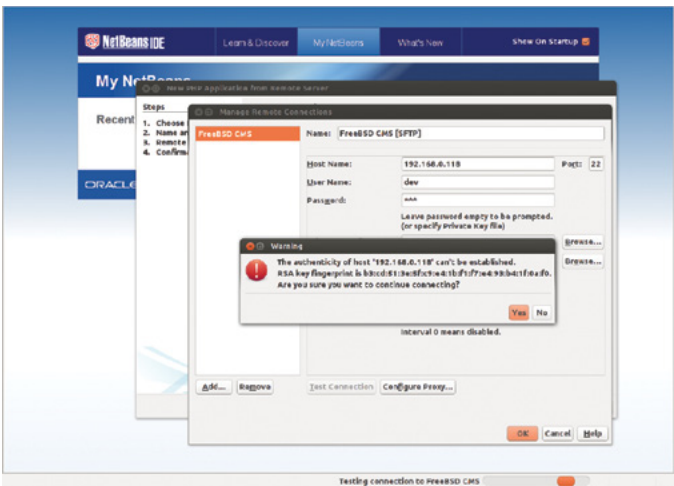


Figure 6. *Testing the connection*

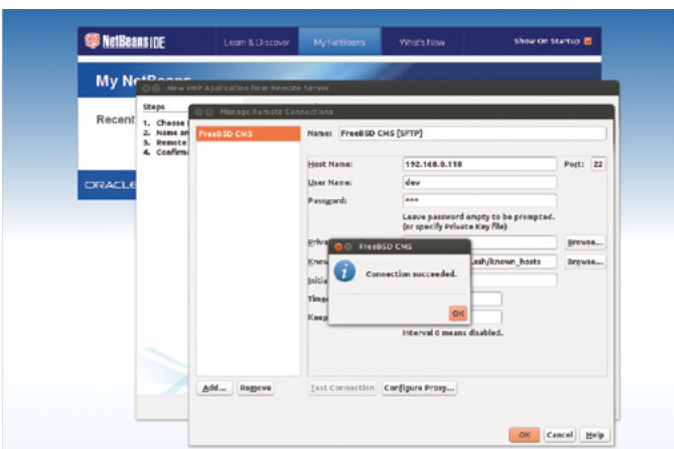


Figure 7. *A successful connection*

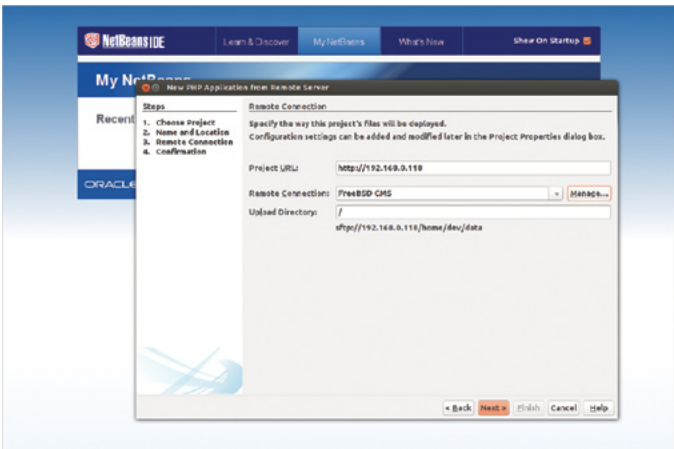


Figure 8. The final settings of the remote project. Replace with your server IP address as required

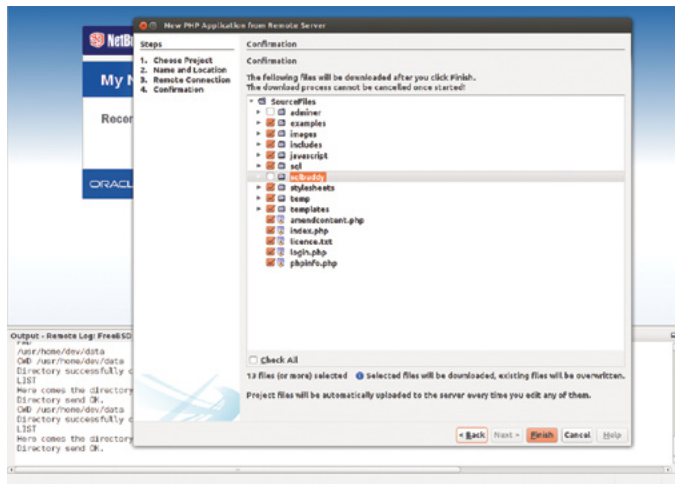


Figure 9. Download the source tree – disable Adminer and sqlbuddy

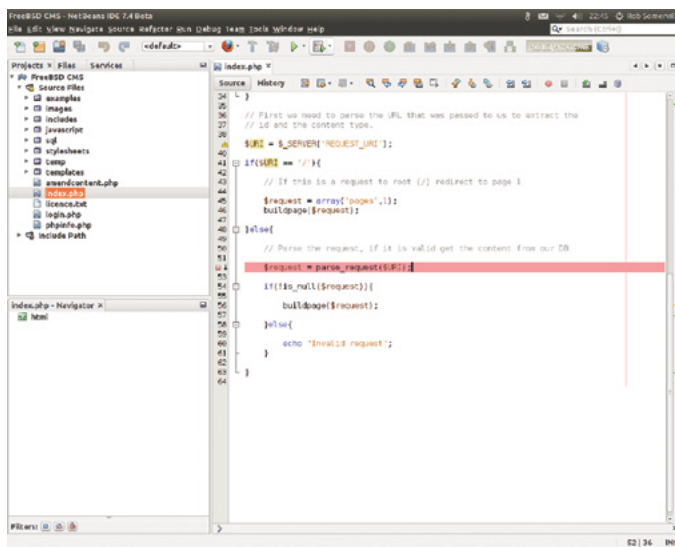


Figure 10. Load *Index.php* click on line 52 and start the debug session by pressing *Ctrl F5*

Now follow the Figures (Figure 2 – 10). If all goes to plan, you should be able to step through your code by pressing F7, and interrogate variables by hovering over them e.g. `$request`. While debugging, the breakpoint line should change colour from pink to green. If it does not there is some mis-communication between Netbeans and the server. See `xdebug.log` for further details.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.



NET OPEN SERVICES IS AN APPLICATION HOSTING COMPANY FOCUSED ON OPEN SOURCE APPLICATIONS MANAGEMENT IN HIGH AVAILABILITY ENVIRONMENT.

NET OPEN SERVICES IS PROUD TO PROVIDE A HIGH QUALITY SERVICE TO OUR CUSTOMERS SINCE 10 YEARS.

OUR EXPERTISE INCLUDES:

- CLOUD COMPUTING, PUBLIC, PRIVATE AND HYBRID CLOUD MANAGEMENT (OPENSTACK, CLOUDSTACK, RED HAT ENTERPRISE VIRTUALIZATION)
- REMOTE MONITORING AND MANAGEMENT 24/7
- NETWORKING AND SECURITY (OPEN BSD, IP TABLE, CHECKPOINT, CISCO,...)
- OS AND APPLICATION MANAGEMENT (FREE BSD, OPEN BSD, SOLARIS, UNIX, LINUX, AIX, MS WINDOWS)
- DATABASE MANAGEMENT (ORACLE, MYSQL, CASSANDRA, NOSQL, MS SQL, SYBASE...)
- MANAGED HOSTING IN CARRIER CLASS DATA CENTERS
- DISASTER RECOVERY



WE PROVIDE SERVICES IN EVERY STEP OF THE PROJECT LIFE, DESIGN, DEPLOYMENT, MANAGEMENT AND EVOLUTIONS. **NETOPENSERVICES** TEAM INCLUDES EXPERIENCED LEADERS AND ENGINEERS IN THE INTERNET SERVER INDUSTRY.

OUR TEAM HAS 15 YEARS OF EXPERIENCE IN DEVELOPING INTERNET INFRASTRUCTURE-GRADE SOLUTIONS AND PROVISIONING INTERNET DATACENTERS AND GLOBAL SERVICE NETWORKS TOGETHER.

WE OFFER EXCEPTIONAL HARDWARE SUPPORT AS SOFTWARE SUPPORT ON UNIX/LINUX AND OPEN SOURCE APPLICATION. **NETOPENSERVICES** DELIVERS THESE CUSTOM-BUILT LINUX AND UNIX SERVERS, AS WELL AS PRECONFIGURED SERVERS AND SCALABLE STORAGE SOLUTIONS, TO OUR CUSTOMERS. WE ALSO OFFER CUSTOM DEVELOPMENT AND ADVANCED-LEVEL UNIX/LINUX CONSULTING SOLUTIONS.

FreeBSD Programming Primer – Part 12

In the final part of our series on programming, we wrap up using the Netbeans Integrated Development Environment to debug and edit our CMS.

What you will learn...

- How to configure a development environment and write HTML, CSS, PHP and SQL code

What you should know...

- BSD and general PC administration skills

Any programmer or developer will freely admit that his or her code is never finished. The best we can hope for is a piece of code that is bug free, reliable and extensible – i.e. we can accommodate future changes easily. Sadly, this is the last part of our programming primer series, and while we have a lot of code (over 3,200 lines excluding external libraries) a lot of further development is required to bring our fledgling CMS up to scratch. While I could carry on and take the project to the point where it is a fully functional CMS, I would not be able personally to support the code and testing cycle in the long term, so it is now time for me to hand this embryonic project over to the community to add the final touches – and squash any inevitable bugs and areas of inefficiency that I have inadvertently included. Rather than me catching the fish, it is now time for you to cast your rod into the deep pool where many fish – (including sharks) – dwell.

In reality, the lesson of Part 12 of the series is probably the hardest in the series – wrapping your head around

Listing 1. `/usr/local/etc/php/xdebug.ini`

```
zend_extension=/usr/local/lib/php/20100525/xdebug.so
xdebug.remote_enable=1
xdebug.remote_host="192.168.0.123"
xdebug.remote_port=9000
xdebug.remote_handler="dbgp"
xdebug.remote_mode=req
xdebug.profiler_enable = 1
xdebug.remote_log=/var/log/xdebug.log
```



Figure 1. Xdebug session initiated in browser

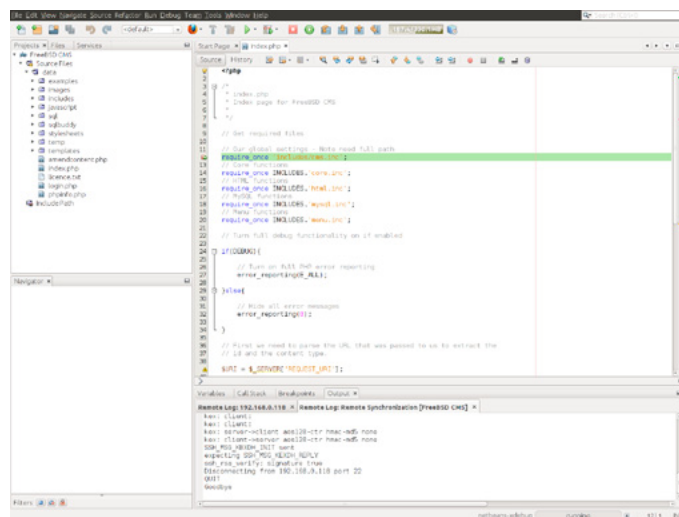


Figure 2. Breakpoint set and Netbeans communicating with remote server

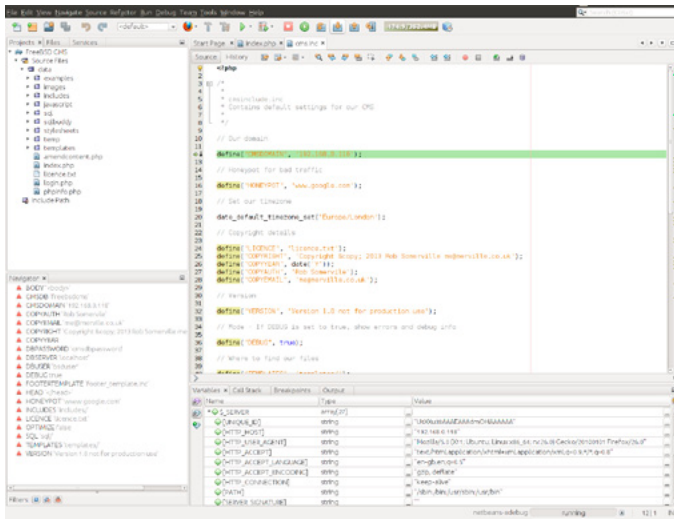


Figure 3. Stepping into CMS.INC

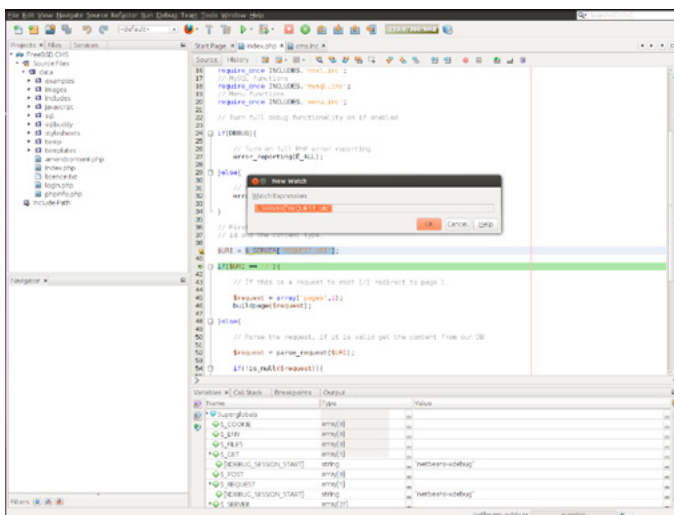


Figure 4. Setting a watch expression

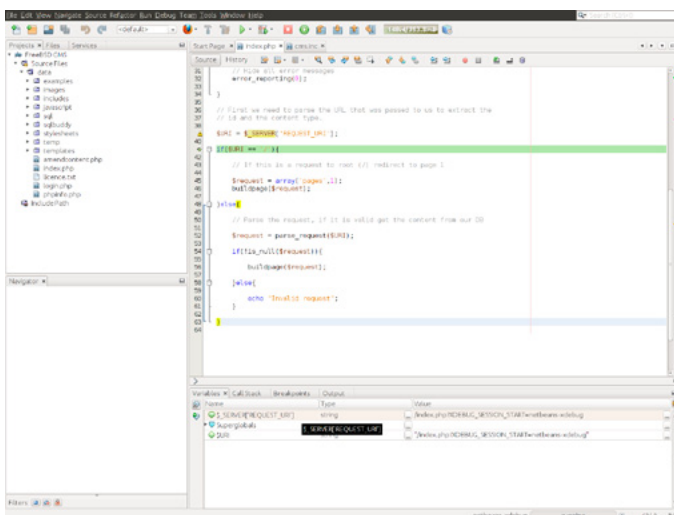


Figure 5. Viewing variables currently set

someone else's code and fixing or developing it. As a developer, this has always been the biggest struggle I have had when faced with maintaining legacy code. It is very easy to "code from scratch" but the demons always lie further down the road. Hopefully, though you will have a head start so that life is a bit easier.

Netbeans and Xdebug

We covered setting up both of these in Part 11. Ensure your xdebug.ini is configured correctly and restart Apache if required /user/local/etc/php/xdebug.ini (Listing 1).

Starting a debugging session

To initiate a real time debugging session, open the FreeBSD CMS project and navigate to line 12 of index.php. Click on the LHS margin next to line 12 to set the breakpoint (which should turn a salmon pink color), and press Ctrl F5 to initiate a debug session. This should open your default browser (in my case Firefox) at the index.php file on the remote server with the parameter XDEBUG_SESSION_START passed to xdebug. This will cause the browser to report "Connecting ..." but no HTML will be parsed as Netbeans is now in control of the program flow. Switch back to Netbeans, and line 12 should be highlighted green – which means we are in debug mode and communicating with the remote server (Figure 1 & 2).

Pressing F7 will walk you through each line of code, and Netbeans will automatically open the first file CMS.INC for you (Figure 3). Continue to press F7 until you come to line 41 of index.php.

Adding a watch

A watch in debugger terms allows you to grab a variable and monitor its value in real time as you step through the code. Highlight `$_SERVER['REQUEST_URI']`, right click and Add Watch. A dialogue box will appear, click OK and the value of this variable will be shown in the lower pane (Figure 4 & 5).

The Call Stack and Breakpoints

The Call stack shows us the code from each of the files that are open. For instance, the function `buildpage()` is contained in `core.inc`. When we reach that point of execution in the code, both `index.php` and `core.inc` will be shown in the call stack. Likewise, all breakpoints are shown in the lower pane (Figure 6 & 7).

Watches and balloon evaluation

If we hover over a variable on the left or right hand side of a statement, we can determine its value. If the value is not shown, ensure this is enabled in the PHP configuration settings (Figure 8 – 10).

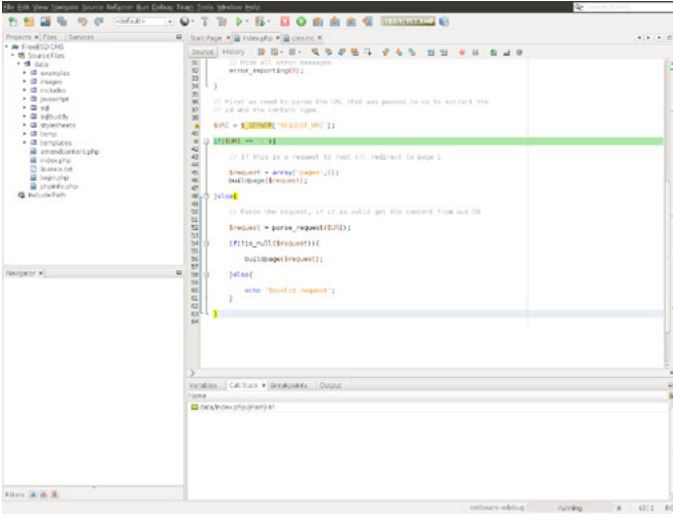


Figure 6. *Viewing the call stack*

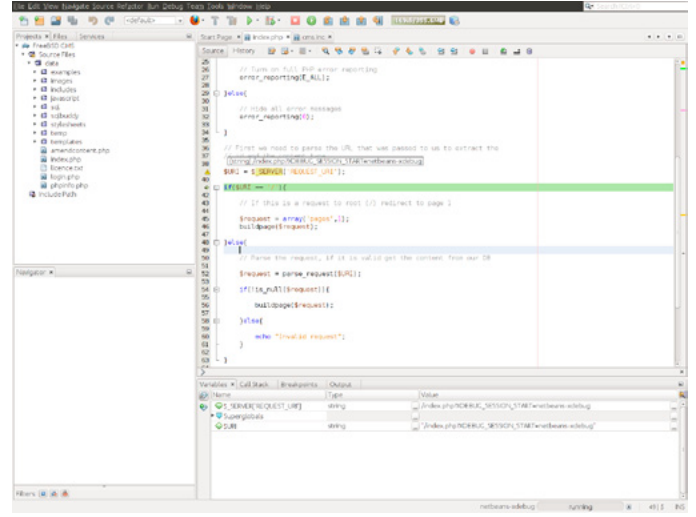


Figure 9. *Viewing the \$URI variable*

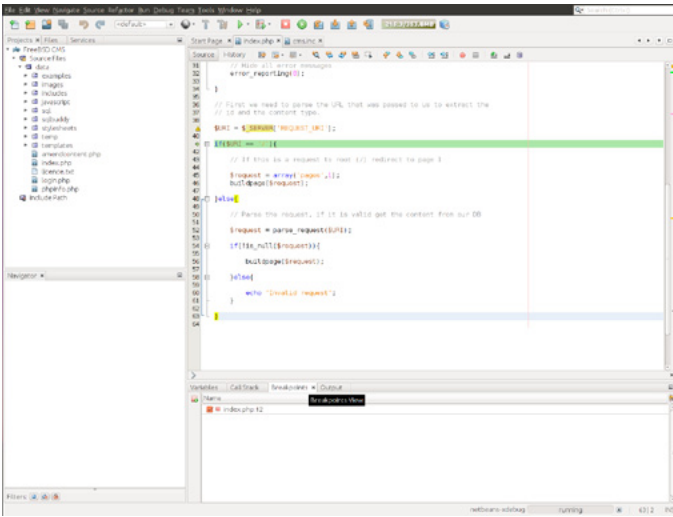


Figure 7. *Viewing breakpoints that are set*

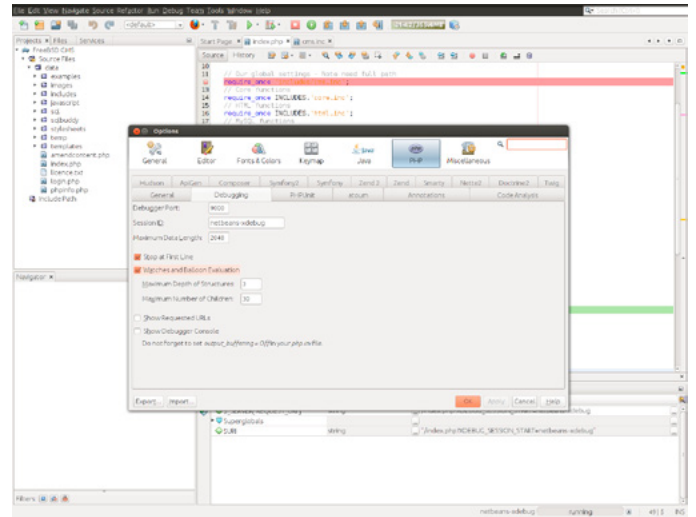


Figure 10. *Setting watches and balloon evaluation*

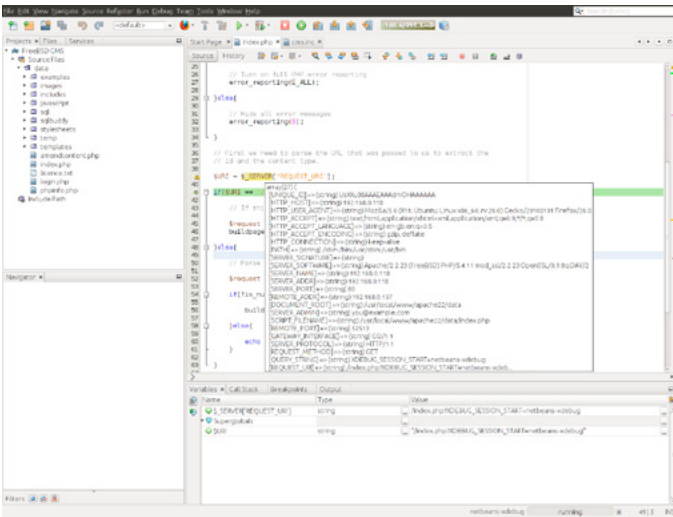


Figure 8. *Real-time balloon watch*

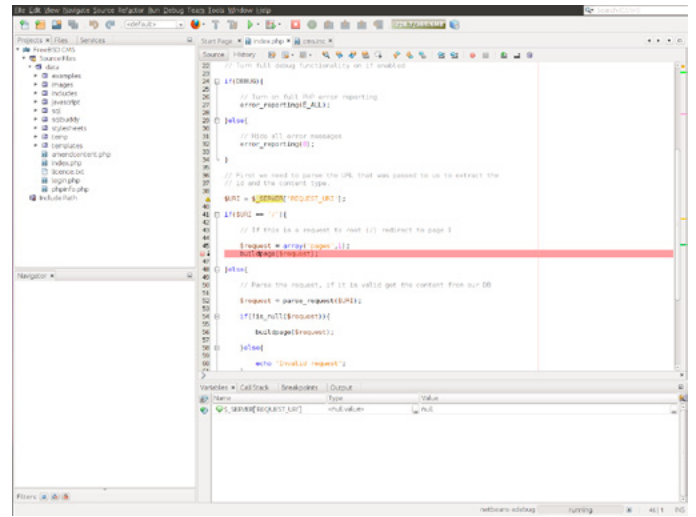


Figure 11. *Setting an additional breakpoint*

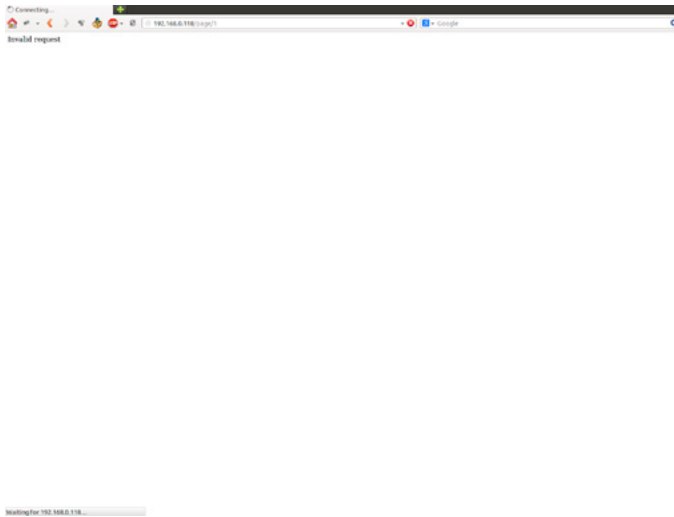


Figure 12. Note the Connecting delay as we step through the code

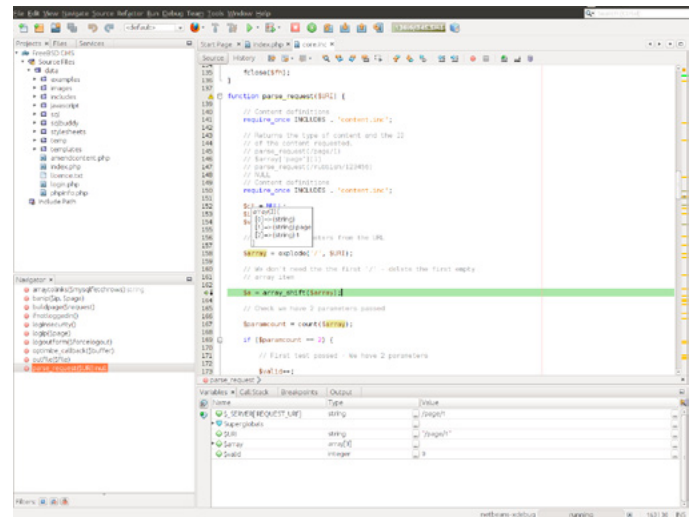


Figure 15. Contents of the \$array variable

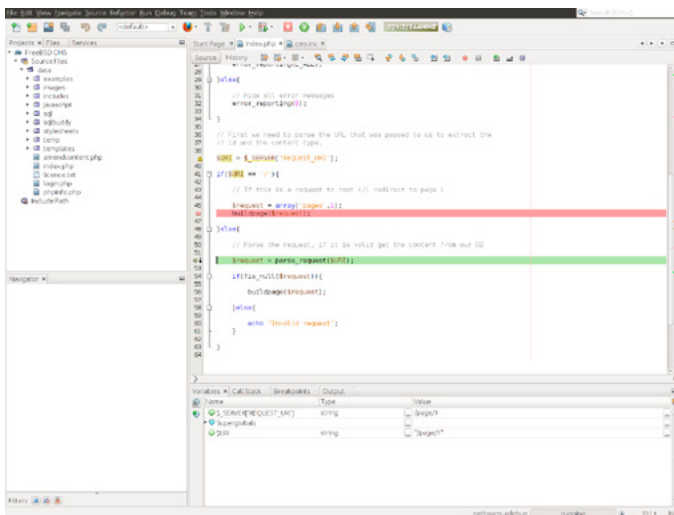


Figure 13. Breakpoints will be jumped over if code is not executed

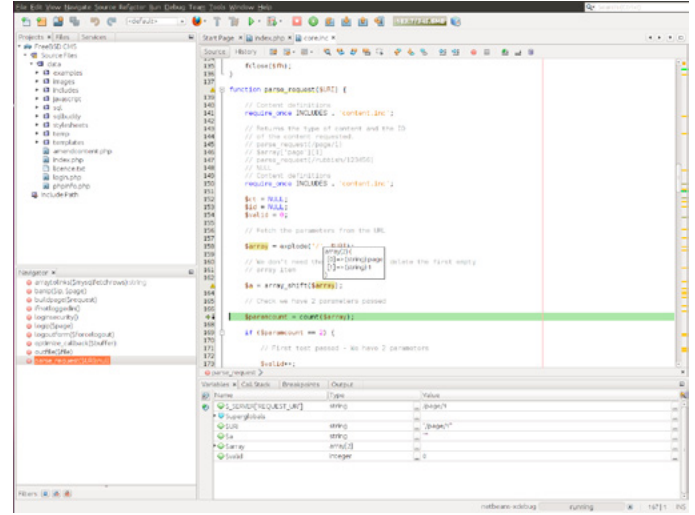


Figure 16. Content of the \$array variable after first element is deleted in code

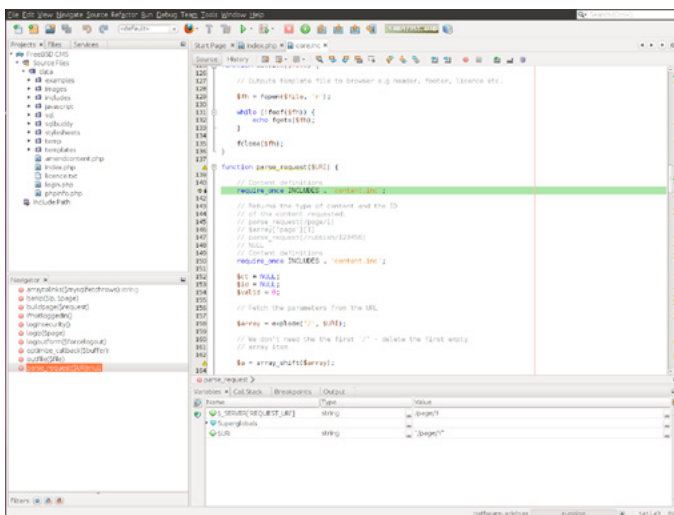


Figure 14. Stepping into the parse_request function

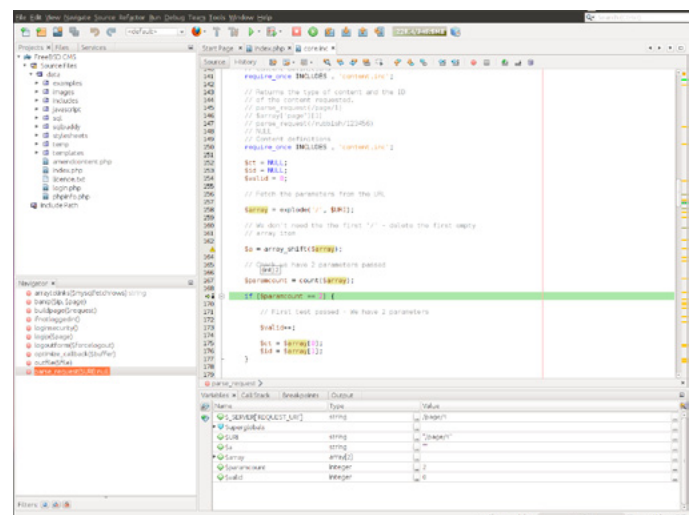


Figure 17. \$paramcount

Program flow

Set a breakpoint at line 46 of index.php. Navigate to ../page/1 in your browser and step through the code using F7. Note that your breakpoint will be ignored as we are not making a request to the home page of the server. Also, the value of \$array will change as we step through the code (Figure 13 – 18). Pressing F5 in Netbeans will allow us to continue until the page is loaded. To prevent Netbeans from stopping at the first line of your code (even if a breakpoint is not set) disable this in the settings (Figure 21).



Figure 18. Once debugger has completed, page is displayed

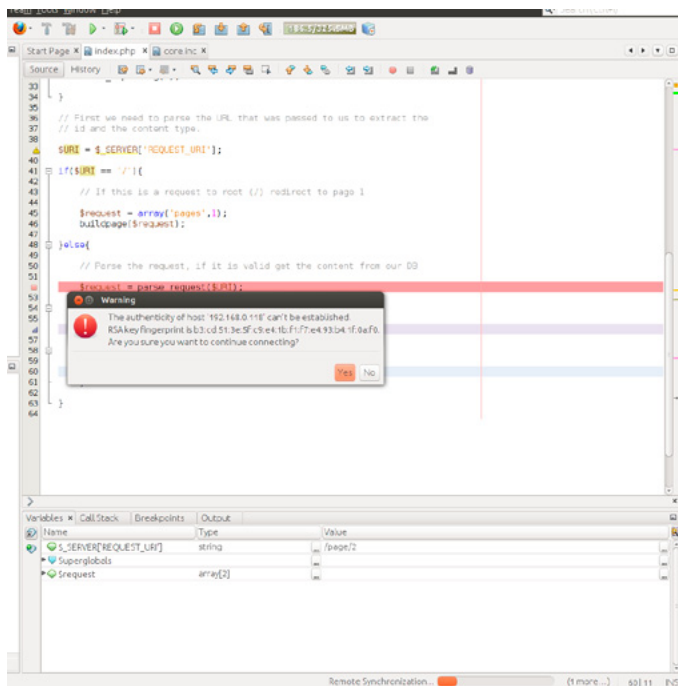


Figure 19. Saving code remotely

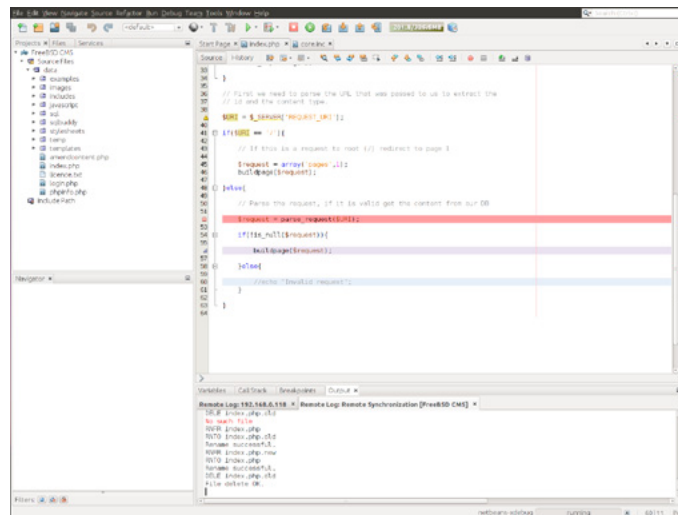


Figure 20. Our modified code

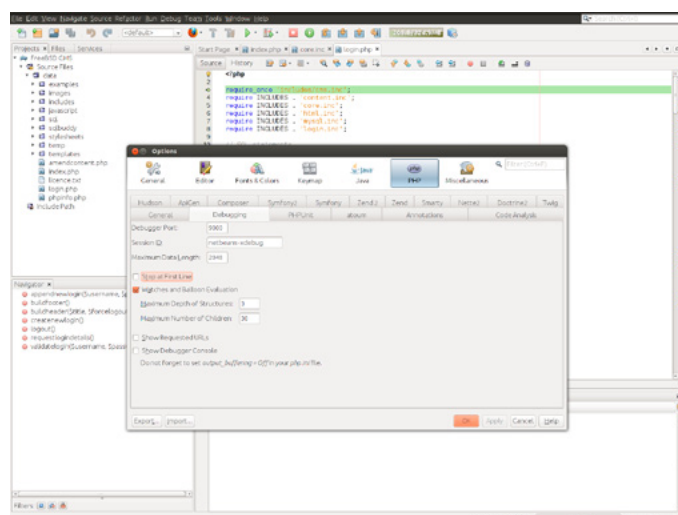


Figure 21. Disabling debugger stopping at the first line

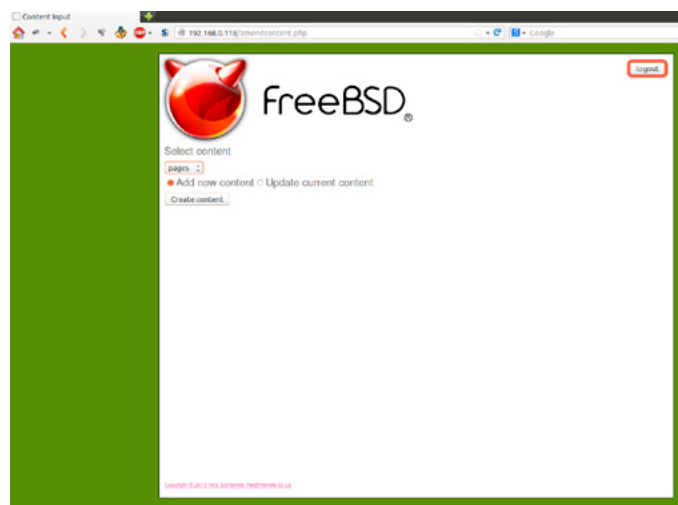


Figure 22. Adding new page content



Figure 23. We have a bug!

Editing and uploading code

Edit line 60 of `index.php` and comment out the `echo` statement. When the file is saved, it should be uploaded to the remote server for you (Figure 19 – 20).

Our first challenge

With the debugger running, log in to the CMS and add some content (Figure 22 – 23). Why is the content not being saved? The answer is at the bottom of the article.

The code

The full code for this project is available at https://github.com/merville/FreeBSD_CMS complete with the database backup which is stored in `DUMP.sql`. Please note that you

Useful links

- Xdebug: <http://xdebug.org>
- Netbeans: <http://php.net/manual>

will have to modify `cms.inc` to meet your own environment. Please note that in its current form this project is not suitable for production use.

So what next?

There are many additional functions our CMS could use, help for user adding content, filtering to check that HTML entered is valid, a facility for uploading photos, additional modules for chat, XML feeds, you name it – the sky is the limit.

The next Howto series will cover image manipulation with the Gimp.

Solution

Line 365 of `amendcontent.php` only displays the changes, they are not committed to the database. For example of this, see the `logip()` function.

ROB SOMERVILLE

Rob Somerville has been passionate about technology since his early teens. A keen advocate of open systems since the mid-eighties, he has worked in many corporate sectors including finance, automotive, airlines, government and media in a variety of roles from technical support, system administrator, developer, systems integrator and IT manager. He has moved on from CP/M and nixie tubes but keeps a soldering iron handy just in case.

a d v e r t i s e m e n t



UPDATE
NOW WITH
STIG
AUDITING

“IN SOME CASES
nipper studio
HAS VIRTUALLY
REMOVED
the **NEED FOR** a
MANUAL AUDIT”
CISCO SYSTEMS INC.

Titania's award winning Nipper Studio configuration auditing tool is helping security consultants and end-user organizations worldwide improve their network security. Its reports are more detailed than those typically produced by scanners, enabling you to maintain a higher level of vulnerability analysis in the intervals between penetration tests.

Now used in over 45 countries, Nipper Studio provides a thorough, fast & cost effective way to securely audit over 100 different types of network device. The NSA, FBI, DoD & U.S. Treasury already use it, so why not try it for free at www.titania.com





Headquarters:
San Jose, CA

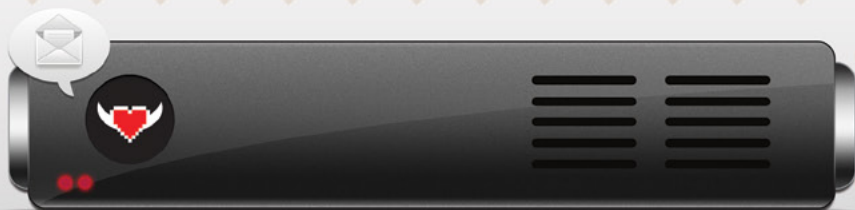


855.GREP.4.IX | Contact Us

99% Compatibility

online now...

IXSYSTEMS AND YOU ARE
THE PERFECT MATCH



SHARED INTERESTS

- ☒ Enterprise Storage Solutions
- ☒ Personalized Customer Service
- ☒ Bold New Information Technology

I'm a

Storage Reseller

In

The EU

Looking for

Storage Solutions to Sell

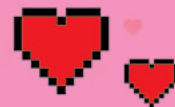
A Technology Partner
More Technical Experience
New Business Opportunities

Visit Today!



iXsystems

Technology Partner Seeking
Resellers/Integrators for
TrueNAS™ Storage Appliance



WWW.IXSYSTEMS.COM/PERFECTMATCH

